PCT/KR 2004/001296
RO/KR 03.06.2004

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2003-0064737
Application Number

출 원 년 월 일 : 2003년 09월 18일
Date of Application SEP 18, 2003

출 원 인 : 한국전자통신연구원
Applicant(s) Electronics and Telecommunications Research In

2004 년 06 월 03 일

특 허 청
COMMISSIONER

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

【서지사항】

【서류명】 특허출원서

【권리구분】 특허

【수신처】 특허청장

【제출일자】 2003.09.18

【발명의 명칭】 라인달 블록 암호 장치와 그 암호화 및 복호화 방법

【발명의 영문명칭】 Apparatus for rijndael block cipher and encryption/decryption method thereof

【출원인】

【명칭】 한국전자통신연구원

【출원인코드】 3-1998-007763-8

【대리인】

【성명】 권태복

【대리인코드】 9-2001-000347-1

【포괄위임등록번호】 2001-057650-1

【대리인】

【성명】 이화익

【대리인코드】 9-1998-000417-9

【포괄위임등록번호】 1999-021997-1

【발명자】

【성명의 국문표기】 이윤경

【성명의 영문표기】 LEE,Yun Kyung

【주민등록번호】 770210-2788011

【우편번호】 770-030

【주소】 경상북도 영천시 문외동 153-1

【국적】 KR

【발명자】

【성명의 국문표기】 박영수

【성명의 영문표기】 PARK,Young Soo

【주민등록번호】 620807-1149123

【우편번호】 302-766

【주소】 대전광역시 서구 탄방동 산호아파트 101동 907호

【국적】 KR

【발명자】
【성명의 국문표기】 김영세
【성명의 영문표기】 KIM,Young Sae
【주민등록번호】 730216-1690618
【우편번호】 700-430
【주소】 대구광역시 중구 대봉2동 719-6
【국적】 KR
【발명자】
【성명의 국문표기】 이상우
【성명의 영문표기】 LEE,Sang Woo
【주민등록번호】 740302-1690216
【우편번호】 302-150
【주소】 대전광역시 서구 만년동 218번지 201호
【국적】 KR
【발명자】
【성명의 국문표기】 전성익
【성명의 영문표기】 JUN,Sung-Ik
【주민등록번호】 620428-1925812
【우편번호】 305-755
【주소】 대전광역시 유성구 어은동 한빛아파트 107동 704호
【국적】 KR
【우선권주장】
【출원국명】 KR
【출원종류】 특허
【출원번호】 10-2003-0038892
【출원일자】 2003.06.16
【증명서류】 첨부
【심사청구】 청구
【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
권태복 (인) 대리인
이화익 (인)

【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	35	면	35,000 원
【우선권주장료】	1	건	26,000 원
【심사청구료】	12	항	493,000 원
【합계】	583,000 원		
【감면사유】	정부출연연구기관		
【감면후 수수료】	304,500 원		

【기술이전】

【기술양도】	희망
【실시권 허여】	희망
【기술지도】	희망

【첨부서류】 1. 요약서·명세서(도면)_1통 2.우선권증명서류 및 동 번역문_1통

【요약서】

【요약】

본 발명은 라인달(Rijndael) 블록 암호를 암호화 및 복호화하기 위한 라운드 연산을 효율적으로 수행하는 연산장치를 포함하는 라인달 블록 암호 장치와 그 암호화 및 복호화 방법에 관한 것이다.

본 발명에 따른 라인달 블록 암호 장치는 고속, 저면적의 암호 프로세서를 요구하는 휴대폰이나 PDA같은 이동단말기 또는 스마트 카드에 탑재함으로써 보안이 필요한 중요한 데이터를 빠른 시간 안에 암호화 및 복호화할 수 있고, 특히 128비트 입력 데이터를 상위 64비트와 하위 64비트로 나누어 라운드 연산을 수행하도록 되어 있다.

따라서, 본 발명은 라인달 블록 암호를 암호화 및 복호화하는데 필요한 시간을 줄이면서 장치의 면적을 감소시킬 수 있다.

【대표도】

도 2

【색인어】

암호 알고리즘, Rijndael, 라인달, 라운드 연산, 블록 암호

【명세서】

【발명의 명칭】

라인달 블록 암호 장치와 그 암호화 및 복호화 방법{Apparatus for rijndael block cipher and encryption/decryption method thereof}

【도면의 간단한 설명】

도 1은 본 발명에 따른 라인달 블록 암호 장치를 도시한 구성도.

도 2는 라운드 연산부를 도시한 구성도.

도 3은 라운드 키 생성부를 도시한 구성도.

도 4는 본 발명에 따른 라인달 블록 암호의 암호화 방법을 도시한 제1타이밍도.

도 5는 본 발명에 따른 라인달 블록 암호의 복호화 방법을 도시한 제1타이밍도.

도 6은 본 발명에 따른 라인달 블록 암호의 암호화 방법을 도시한 제2타이밍도.

도 7은 본 발명에 따른 라인달 블록 암호의 복호화 방법을 도시한 제2타이밍도.

도 8은 본 발명에 따른 라인달 블록 암호의 암호화 방법을 도시한 제3타이밍도.

도 9는 본 발명에 따른 라인달 블록 암호의 복호화 방법을 도시한 제3타이밍도.

<도면의 주요부분에 대한 부호의 설명>

100: 라운드 연산부 110: 라운드 키 생성부

120: 시프트/역시프트_로우 변환부 121: 제1다중화기

130: 대치/역대치 변환부 140: 제1역다중화기

150: 믹스/역믹스컬럼 변환부 160: 제2역다중화기

170: 라운드 키 더하기 변환부 180: 제3역다중화기

200: 버스 300: 라운드 연산 제어부

400: 64비트 데이터 레지스터 500: 128비트 데이터 레지스터

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<18> 본 발명은 라인달(Rijndael) 블록 암호 장치와 그 암호화 및 복호화 방법에 관한 것으로서, 보다 상세히는 휴대폰이나 PDA, 스마트 카드 등에 탑재함으로써 보안이 필요한 중요한 데이터를 빠른 시간 안에 암호화 및 복호화할 수 있도록 된 라인달 블록 암호 장치와 그 암호화 및 복호화 방법에 관한 것이다.

<19> 라인달 알고리즘은 벨기에의 암호개발자인 존 데몬(Joan Daemen)과 빈센트 리즈멘(Vincent Rijmen)에 의해 개발된 후 2000년 10월경 미국의 국립 표준 기술원(NIST; National Institute of Standards and Technology)에서 새로운 첨단 암호 표준(AES; Advanced Encryption Standard)으로 선정한 대칭 비밀키 암호화 알고리즘이다.

<20> 라인달 알고리즘은 SPN(Substitution-Permutation Network) 구조의 가변 블록길이를 지원하고, 각 블록길이에 대해 128비트, 192비트, 256비트 키를 사용할 수 있다.

<21> 라인달 알고리즘의 라운드 수는 키 길이에 의해 결정되고, 128비트 블록을 사용하는 경우 128, 192, 256비트 키에 대해 각각 10, 12, 14라운드를 사용하도록 권고되고 있다.

2> 현재 라인달 알고리즘은 128비트 키를 사용하여도 안전성에는 문제가 없는 것으로 알려져 있으며, 이에 따라서 128비트 길이의 키를 이용한 라인달 알고리즘의 하드웨어 구현에 관하여 연구가 진행되고 있는 실정이다.

3> 라인달 알고리즘은 라운드 연산의 반복에 의해서 라인달 블록 암호화/복호화용 데이터를 암호화 또는 복호화하며, 특히 SPN 구조의 가변 블록길이를 지원하도록 되어있기 때문에 블록 암호의 암호화 과정과 복호화 과정이 다르다. 통상적으로, 라인달 블록 암호의 암호화 과정을 위한 라운드 연산은 대치(Substitution), 시프트_로우(Shift_Row), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 네 가지 변환으로 이루어지고, 복호화 과정을 위한 라운드 연산은 역시프트_로우(Inverse Shift_Row), 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 네 가지 변환으로 이루어진다. 이들 변환을 수행하는 방법에 따라 라인달 블록 암호에 대한 라운드 연산에 소요되는 시간과 사용하는 하드웨어 리소스에 차이가 있으며, 이는 더 나아가 라인달 암호 프로세서의 성능과 직결되는 문제가 된다. 따라서 라운드 연산의 구현에 필요한 하드웨어 리소스의 양과 라운드 연산 수행에 소요되는 시간을 줄이는 것이 중요하다.

【발명이 이루고자 하는 기술적 과제】

24> 따라서, 본 출원인은 상기한 바와 같은 관점에서 라인달 블록 암호를 암호화 및 복호화하기 위한 라운드 연산을 효율적으로 수행하는 연산장치를 포함하는 라인달 블록 암호 장치와 그 암호화 및 복호화 방법을 개발하게 되었다.

5> 본 발명의 목적은 고속, 저면적의 암호 프로세서를 요구하는 휴대폰이나 PDA같은 이동단말기 또는 스마트 카드에 탑재함으로써 보안이 필요한 중요한 데이터를 빠른 시간 안에 암호화 및 복호화할 수 있도록 된 라인달 블록 암호 장치와 그 암호화 및 복호화 방법을 제공하는데 있다.

【발명의 구성】

26> 본 발명은 M비트의 입력데이터 및 N비트의 입력키를 가지며, 시프트_로우(Shift_Row), 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정을 포함하는 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 암호화하는 라인달 블록 암호화 장치에 있어서, 적어도 상기 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하는 라운드 연산 실행부와 상기 라운드 키 더하기 과정에 라운드키를 제공하기 위해 라운드키를 생성하는 라운드키생성부를 포함하는 라운드 연산부; 상기 라운드 연산부에 의해 수행되는 라운드 연산을 제어하는 라운드 연산 제어부; 및 상기 라운드 연산부에 의해 매 라운드의 중간 단계에서 생성되는 M/m 비트의 중간 데이터 및 매 라운드의 종료단계에서 생성되는 M 비트의 데이터를 저장하기 위한 데이터기억부를 포함하는 것을 특징으로 하는 라인달 블록 암호화 장치를 제공한다.

27> 또한, 본 발명은 M비트의 입력데이터 및 N비트의 입력키를 가지며, 역시프트_로우(Inverse Shift_Row), 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정을 포함하는 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 복호화하는 라인달 블록 복호화 장치에 있어서, 적어도 상기

역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하는 라운드 연산 실행부와 상기 라운드 키 더하기 과정에 라운드키를 제공하기 위해 라운드키를 생성하는 라운드키생성부를 포함하는 라운드 연산부; 상기 라운드 연산부에 의해 수행되는 라운드 연산을 제어하는 라운드 연산 제어부; 및 상기 라운드 연산부에 의해 매 라운드의 중간단계에서 생성되는 M/m 비트의 중간 데이터 및 매 라운드의 종료단계에서 생성되는 M 비트의 데이터를 저장하기 위한 데이터기억부를 포함하는 것을 특징으로 하는 라인달 블록 복호화 장치를 제공한다.

28> 또한, 본 발명은 M M비트의 입력데이터 및 N비트의 입력키를 가지며, 시프트_로우(Shift_Row), 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정을 포함하는 암호화용 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 암호화하거나, 역시프트_로우(Inverse Shift_Row), 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정을 포함하는 복호화용 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 복호화하기 위한 라인달 블록 암복호화 장치에 있어서, 암호화 모드에서는, 적어도 상기 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하고, 복호화 모드에서는, 적어도 상기 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하는 라운드 연산 실행부와 상기 라운드 키 더하기 과정에 라운드키를 제공하기 위해 라운드키를 생성하는 라운드키생성부를 포함하는 라운드 연산부; 상기 라운드 연산부에 의해 수행되는 라운드 연산을 제어하는 라운드 연산 제어부; 및 상기 라운드 연산부에 의해 매 라운드의 중간단계에서 생성되는 M/m 비트의 중간

데이터 및 매 라운드의 종료단계에서 생성되는 M 비트의 데이터를 저장하기 위한 데이터기억부를 포함하는 것을 특징으로 하는 라인달 블록 암복호화 장치를 제공한다.

<29> 또한, 본 발명은 M비트의 입력데이터와 N비트의 입력키를 입력받아, 소정 회수만큼 라운드연산을 수행하는 라인달 블록 암호화 방법에 있어서, 상기 방법은, 이전 라운드로부터의 M 비트 데이터를 시프트_로우(Shift_Row)변환을 가하고, 선택신호에 대응하는 M/m 비트(m은 2, 3 또는 4) 데이터만을 다음 단계로 출력하는 시프트_로우(Shift_Row)변환단계; 상기 M/m 비트 데이터에 대하여 대치(Substitution)변환을 실시하는 대치(Substitution)변환단계; 상기 M/m 비트 데이터에 대하여 믹스컬럼(MixColumn)변환을 실시하는 믹스컬럼(MixColumn)변환단계; 및 상기 M/m 비트 데이터에 동일한 크기의 라운드 키를 더하는 라운드 키 더하기(Add Round key)단계를 m개의 M/m비트 데이터 모두에 대해 수행하는 것을 하나의 라운드로 하는 라운드 연산단계; 및 상기 라운드키 더하기 단계에 라운드 키를 제공하기 위해 라운드키를 생성하는 라운드키 생성단계를 포함하는 것을 특징으로 하는 라인달 블록 암호화 방법을 제공한다.

<30> 또한, 본 발명은 M비트의 입력데이터와 N비트의 입력키를 입력받아, 소정 회수만큼 라운드연산을 수행하는 라인달 블록 복호화 방법에 있어서, 상기 방법은, 이전 라운드로부터의 M 비트 데이터를 역시프트_로우(Inverse Shift_Row)변환을 가하고, 선택신호에 대응하는 M/m 비트(m은 2, 3 또는 4) 데이터만을 다음 단계로 출력하는 역시프트_로우(Inverse Shift_Row)변환단계; 상기 역시프트_로우 변환된 M/m 비트 데이터에 대하여 역대치(Inverse Substitution)변환을 실시하는 역대치(Inverse Substitution)변환단계; 상기 역대치 변환된 M/m 비트 데이터에 동일한 크기의 라운드 키를 더하는 라운드 키 더하기(Add Round key)단계; 및 상기 라운드 키를 더한 M/m 비트 데이터에 대하여 역믹스컬럼(Inverse MixColumn)변환을 실시하는 역믹스컬럼(Inverse MixColumn)변환단계를 m개의 M/m비트 데이터 모두에

대해 수행하는 것을 하나의 라운드로 하는 라운드 연산단계; 및 상기 라운드키 더하기 단계에 라운드 키를 제공하기 위해 라운드키를 생성하는 라운드키 생성단계를 포함하는 것을 특징으로 하는 라인달 블록 복호화 방법을 제공한다.

<1> 본 발명에 따른 라인달 블록 암/복호화 장치에서는 라인달 블록 암호화/복호화용 입력 데이터의 크기가 128비트, 192비트, 256비트 중 어느 하나일 수 있으며, 이를 일정한 크기로 잘라 라인달 블록 암호화 또는 복호화 연산과정을 거치게 되므로, 상기 연상과정에 요구되는 하드웨어 자원을 절약할 수 있다. 여기서 입력데이터를 쪼갤 수 있는 최대 수를 4이하로 하는 것이 바람직한데, 이는 라운드연산이 4단계(예컨대, 암호화의 경우에는 시프트_로우(Shift_Row); 대치(Substitution); 믹스컬럼(MixColumn); 및 라운드 키 더하기(Add Round key)의 4단계)로 이루어지는 것과 관련이 된다. 본 발명에 따른 암복호화 장치는 처리가 요구되는 데이터 및 키의 크기와 하드웨어 및 처리시간에 따른 제약조건을 모두 만족할 수 있도록, 데이터를 나누는 개수를 정할 수 있을 것이다. 그러나 일반적으로 입력데이터를 4개보다 더 많게 나누어서 처리하려면, 처리시간이 늘어남은 물론이고, 추가의 레지스터가 필요하게 될 수 있으므로 바람직하지 않다.

<32> 또한, 상기 라운드 연산을 수행하면서 동시에 128비트, 192비트, 256비트 중 어느 하나일 수 있는 입력 키(key)를 이용하여 상기 라운드 연산(즉, 라운드 키 더하기)에 필요한 라운드 키(RK)를 생성하도록 되어 있다.

<33> 본 발명에 따른 암호화/복호화 방법 및 장치에서는 입력데이터가 128비트, 192비트, 256비트 중 어느 하나인 것이 일반적이며, 입력키도 128비트, 192비트, 256비트 중 어느 하나인 것이 일반적이다.

이하에서는 본 발명에 따른 실시예들을 첨부한 도면을 참조하여 상세히 설명하기로 한다. 설명의 편의를 위해 입력데이터와 입력키가 모두 128비트인 경우에 대해서만 설명을 하고 있지만, 입력데이터 및 입력키의 크기가 달라지는 경우에도 마찬가지로 적용이 될 수 있다. 특별히, 이하의 실시예에서는 128비트 크기의 입력데이터를 2개의 64비트 데이터로 나누어 처리하는 것에 대해 상술하고 있다.

도 1 및 도 2는 본 발명에 따라 128비트 입력 데이터를 128비트 입력 키(key)를 이용하여 암호화 또는 복호화하는 라인달 블록 암/복호화 장치의 실시예이다.

도 1을 참조하면, 라운드 연산부(100)는 라인달 블록 암호화/복호화용 버스(200)를 통해 암호 또는 복호연산 시작신호(start)와 모드신호(mode)가 들어온 후 라운드 연산 제어부(300)로부터 라운드 연산 시작신호(Round_start)와 라운드 번호신호(Round_number) 및 매 라운드 연산마다 상기 128비트 입력 데이터를 상위 64비트와 하위 64비트로 나누어 선택하기 위한 비트 선택신호(sel)가 들어오면 이때부터 모드신호의 값에 따라서 128비트 입력 키(key)를 암호화용 또는 복호화용 128비트 라운드 키(RK)로 변환하여 저장한다.

상기 라운드 연산부(100)는 상기 모드신호의 값이 '0'을 나타내면 128비트 입력 데이터를 상위 64비트와 하위 64비트로 나누어 각각 시프트_로우, 대치, 믹스컬럼, 라운드 키 더하기의 변환 과정으로 이루어진 라운드 연산을 수행하여 상기 128비트 입력 데이터를 암호화한다.

상기 라운드 연산부(100)는 상기 모드신호의 값이 '1'을 나타내면, 128비트 입력 데이터를 상위 64비트와 하위 64비트로 나누어 각각 역시프트_로우, 역대치, 라운드 키 더하기, 역믹스컬럼의 변환 과정으로 이루어진 라운드 연산을 수행하여 상기 128비트 입력 데이터를 복호화한다.

> 라운드 연산 제어부(300)는 상기 버스(100)를 통해 암호 또는 복호연산 시작신호와 모드 신호가 들어오면 이때부터 라운드 연산 시작신호(Round_start)와 라운드 번호신호 (Round_number) 및 매 라운드 연산마다 상기 128비트 입력 데이터를 상위 64비트와 하위 64비트로 나누어 선택하기 위한 비트 선택신호(sel)를 상기 라운드 연산부(100)로 전달하여 상기 라운드 연산부(100)의 라운드 연산 작동을 제어한다.

)> 64비트 데이터 레지스터(400)는 상기 라운드 연산부(100)에 의해 수행되는 매 라운드 연산 중에 생성되는 상위 64비트 입력 데이터의 중간 암호 또는 복호 데이터를 저장한다.

1> 128비트 데이터 레지스터(500)는 상기 라운드 연산부(100)에 의해 수행되는 매 라운드 연산 중에 생성되는 하위 64비트 입력 데이터의 중간 암호 또는 복호 데이터를 하위 64비트로 저장하고, 매 라운드 연산 결과로 생성된 상기 상위 64비트 데이터 및 상기 하위 64비트 데이터를 각각 상위 및 하위 64비트로 저장하여, 128비트 데이터를 다음 라운드의 입력데이터로 제공한다.

42> 도 1에 있어서, 매 라운드 연산 중에 생성되는 128비트 입력 데이터의 중간 암/복호 데이터, 매 라운드 종료시의 암/복호 데이터와 마지막 라운드 연산 결과로 생성된 최종 암호 또는 복호 데이터를 저장하는 복수의 데이터 레지스터로서 64비트 데이터 레지스터(400)와 128비트 데이터 레지스터(500)를 사용하지만, 다른 실시예로서 192비트 또는 256비트 입력 데이터를 128비트, 192비트, 256비트 입력 키(key) 중 어느 하나를 이용하여 암호화 또는 복호화하는 경우, 상기 복수의 데이터 레지스터는 다른 구조로 구현된다.

:43> 예컨대, 192비트 입력 데이터를 암호화 또는 복호화하는 경우, 상기 라운드 연산부(100)에 의해 수행되는 매 라운드 연산 중에 생성되는 상위 64비트 입력 데이터와 그 다음 64비트 입력 데이터의 중간 암호 또는 복호 데이터를 각각 저장하는 2개의 64비트 데이터 레지스터(또

는 1개의 128비트 데이터 레지스터)와, 상기 라운드 연산부(100)에 의해 수행되는 매 라운드 연산 중에 생성되는 하위 64비트 입력 데이터의 중간 암호 또는 복호 데이터를 하위 64비트로 저장하고, 매 라운드 연산 결과로 생성된 상기 2개의 64비트 데이터 레지스터에 각각 저장된 암호 또는 복호 데이터를 상위 64비트와 그 다음 64비트로 저장하는 192비트 데이터 레지스터로 구현될 수 있다.

4> 본 발명의 구현에 필요한 레지스터 공간의 크기(R)를 보다 일반적으로 표현하면, R = M(m-1)/m + M = (2m-1)M/m[여기서 M 은 입력데이터의 비트수, m 은 나누어져서 처리되는 데이터의 개수임]로 나타낼 수 있다. 상기 실시예를 예로 들어 보면, 128비트가 64비트 2개로 나누어져서 처리되므로, R = 128*1/2 + 128 = 192비트가 된다. 또한, 192비트가 3개로 나누어져서 처리되는 경우라면, R = 192*2/3 + 192 = 320비트가 되는 것을 알 수 있다. 이는 일반적인 파이프라인 방식을 이용하는 것에 비해 M/m 만큼의 레지스터 공간을 절약할 수 있는 효과를 준다.

45> 도 2를 참조하면, 상기 라운드 연산부(100)의 라운드 키 생성부(110)는 상기 라운드 연산 제어부(300)로부터 라운드 연산 시작신호와 라운드 번호신호가 들어오면 상기 버스(200)를 통해 들어온 모드신호의 값에 따라서 128비트 입력 키(key)를 암호화용 또는 복호화용 128비트 라운드 키(RK)로 변환하여 내부의 128비트 라운드 키 레지스터에 저장한다.

46> 상기 라운드 연산부(100)의 시프트/역시프트_로우 변환부(120)는 상기 라운드 연산 제어부(300)로부터 라운드 연산 시작신호와 비트 선택신호가 들어오면 상기 버스(200)를 통해 들어온 모드신호의 값에 따라서 128비트의 이전 라운드의 데이터(단, 제1라운드에서는 128비트 입력데이터)를 바이트 시프트 변환한 후, 상기 비트 선택신호의 값에 따라서 출력이 제어되는 내

부의 제1다중화기(121)를 통해 시프트 변환된 128비트 데이터를 상위 64비트(비트선택신호="1"인 경우)와 하위 64비트(비트선택신호="0"인 경우)로 순차적으로 나누어 출력한다.

<7> 상기 라운드 연산부(100)의 대치/역대치 변환부(130)는 상기 시프트/역시프트_로우 변환부(120)로부터 출력되는 상위 64비트 데이터 또는 하위 64비트 데이터를 한 바이트의 입력에 대해서 한 바이트의 출력을 내보내는 대치 박스(S-box) 또는 역대치 박스(SI-box)를 이용하여 대치 또는 역대치 연산한다.

<8> 상기 라운드 연산부(100)의 제1역다중화기(140)는 상기 대치/역대치 변환부(130)로부터 출력되는 상위 64비트 데이터 또는 하위 64비트 데이터를 상기 모드신호의 값에 따라서 암호 출력단(0)과 복호 출력단(1) 중 어느 하나로 출력한다.

<9> 상기 라운드 연산부(100)의 믹스/역믹스컬럼 변환부(150)는 상기 제1역다중화기(140)의 암호 출력단(0)을 통해 입력되는 상위 64비트 데이터 또는 하위 64비트 데이터를 믹스컬럼 변환하거나 라운드 키 더하기 변환된 상위 64비트 데이터 또는 하위 64비트 데이터를 역믹스컬럼 변환한다.

<50> 상기 라운드 연산부(100)의 제2역다중화기(160)는 상기 믹스/역믹스컬럼 변환부(150)로부터 출력되는 상위 64비트 데이터 또는 하위 64비트 데이터를 상기 모드신호의 값에 따라서 암호 출력단(0)과 복호 출력단(1) 중 어느 하나로 출력한다.

<51> 상기 라운드 연산부(100)의 라운드 키 더하기 변환부(170)는 상기 제1역다중화기(140)의 복호 출력단(1) 또는 상기 제2역다중화기(160)의 암호 출력단(0)을 통해 입력되는 상위 64비트 데이터 또는 하위 64비트 데이터를 상기 라운드 키 생성부(110)로부터 출력되는 암호화용 또는 복호화용 128비트 라운드 키(RK)와 라운드 키 더하기 변환한다.

2> 상기 라운드 연산부(100)의 제3역다중화기(180)는 상기 라운드 키 더하기 변환부(170)로부터 출력되는 상위 64비트 데이터 또는 하위 64비트 데이터를 상기 모드신호의 값에 따라서 암호 출력단(0)과 복호 출력단(1) 중 어느 하나로 출력한다.

3> 여기서, 설명한 암/복호화 장치는 암호화모드와 복호화모드에서 사용되는 하드웨어 리소스를 공유하여, 하드웨어가 차지하는 면적을 최소화하고, 128비트의 데이터를 처리하면서도 64비트처리용 모듈을 사용할 수 있게 하며, 그렇게 하면서도 처리시간이 거의 늘어나지 않게 된다(연산처리 타이밍에 대해서는 후술함).

54> 도 3은 본 발명에 따른 라운드 키 생성부의 일실시예를 나타낸다. 여기서, 라운드 키 생성부(110)의 128비트 프리 키 레지스터(111)는 버스(200)를 통해 들어온 128비트 입력 키(key)를 암호화용 또는 복호화용 128비트 라운드 키(RK)로 변환하기 위한 프리 키(prekey)로 저장하고, 매 라운드 연산후 생성된 128비트 라운드 키(RK)를 다음 라운드 연산에서 사용되는 라운드 키(RK)를 생성하기 위한 프리 키(prekey)로 저장한다.

55> 상기 라운드 키 생성부(110)의 128비트 키 레지스터(111a)는 매 라운드 연산을 위한 암호화용 또는 복호화용 128비트 라운드 키(RK)를 저장한다. 도 3에서 상기 128비트 라운드 키 레지스터(111a)에 저장되는 128비트 라운드 키(RK)는 매 라운드 연산 종료 후 상기 128비트 프리 키 레지스터(111)에 백업되어 다음 라운드 연산에서 이전 라운드의 라운드 키(prekey)로 사용된다.

<56> 상기 라운드 키 생성부(110)의 상수저장부(112)는 상기 라운드 연산 제어부(300)로부터 들어오는 라운드 번호신호가 나타내는 라운드가 몇 번째 라운드인지에 따라서 결정되는 상수값(Rcon)을 저장한다. 상기 상수저장부(112)로는 ROM을 사용하는 것이 바람직하다.

7> 상기 라운드 키 생성부(110)의 제2다중화기(113)는 상기 버스(200)를 통해 들어온 모드 신호의 값에 따라서 출력이 제어되어 상기 128비트 프리 키 레지스터(111)와 상기 128비트 라운드 키 레지스터(111a)로부터 입력되는 암호화용 또는 복호화용 32비트 키 중 어느 하나를 선택하여 출력한다.

8> 상기 라운드 키 생성부(110)의 시프트기(114)는 상기 제2다중화기(113)를 통해 입력되는 32비트 키를 한 바이트 왼쪽으로 순환 시프트한다.

9> 상기 라운드 키 생성부(110)의 대치변환기(115)는 대치 연산을 수행하는 대치 박스들로 이루어져 상기 시프트기(114)에 의해 순환 시프트된 32비트 키를 대치 변환한다.

0> 상기 라운드 키 생성부(110)의 제1XOR연산기(116)는 상기 대치변환기(115)로부터 출력되는 32비트 키 중 최상위 한 바이트 값과 상기 상수저장부(112)에 저장된 상수값과의 XOR연산을 수행한다.

61> 상기 라운드 키 생성부(110)의 라운드 XOR연산부(117)는 상기 제1XOR연산기(116)의 출력 값과 대치변환기(115)의 최상위 한 바이트 값을 제외한 나머지 24비트 값이 합쳐진 32비트 값과, 상기 128비트 프리 키 레지스터(111)에 저장되는 이전 라운드의 128비트 라운드 키 (prekey) 및, 상기 128비트 라운드 키 레지스터(111a)에 저장되는 새로운 라운드의 128비트 라운드 키(RK)를 이용하는 XOR연산을 수행하여 상기 128비트 라운드 키 레지스터(111a)에 저장되는 암호화용 또는 복호화용 128비트 라운드 키(RK)를 라운드 연산의 매 라운드마다 새롭게 생성한다.

<62> 상기 라운드 XOR연산부(117)의 제2XOR연산기(118)는 상기 제1XOR연산기(116)의 출력 값과 대치변환기(115)의 최상위 한 바이트 값을 제외한 나머지 24비트 값이 합쳐진 32비트 값과,

이전 라운드의 128비트 라운드 키 중 최상위 32비트(PK0)를 XOR연산하여 새로운 라운드의 암호화용 또는 복호화용 128비트 라운드 키 중 최상위 32비트 라운드 키(RK0)를 생성한다.

<3> 상기 라운드 XOR연산부(117)의 제3XOR연산기(118a)는 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 32비트 다음의 95번째부터 64번째까지의 32비트 라운드 키(PK1)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 95번째부터 64번째까지의 32비트 라운드 키(RK1)를 생성한다.

<4> 상기 제3XOR연산기(118a)는 이전 라운드의 128비트 라운드 키의 127번째부터 96번째까지의 최상위 32비트 라운드 키(PK0)와 그 다음 95번째부터 64번째까지의 32비트 라운드 키(PK1)를 XOR연산하여 새로운 라운드의 복호화용 128비트 라운드 키 중 95번째부터 64번째까지의 32비트 라운드 키(RK1)를 생성한다.

<5> 상기 라운드 XOR연산부(117)의 제3다중화기(119)는 상기 버스(200)를 통해 들어온 모드 신호의 값에 따라서 출력이 제어되어 상기 제3XOR연산기(118a)의 입력신호를 선택적으로 결정한다.

<6> 상기 라운드 XOR연산부(117)의 제4XOR연산기(118b)는 새로운 라운드의 128비트 라운드 키 중 95번째부터 64번째까지의 32비트 라운드 키(RK1)와 이전 라운드의 128비트 라운드 키 중 63번째부터 32번째까지의 32비트 라운드 키(PK2)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 63번째부터 32번째까지의 32비트 라운드 키(RK2)를 생성한다.

<7> 상기 제4XOR연산기(118b)는 이전 라운드의 128비트 라운드 키의 95번째부터 64번째까지의 32비트 라운드 키(PK1)와 그 다음 63번째부터 32번째까지의 32비트 라운드 키(PK2)를 XOR연

산하여 새로운 라운드의 복호화용 128비트 라운드 키 중 63번째부터 32번째까지의 32비트 라운드 키(RK2)를 생성한다.

<68> 상기 라운드 XOR연산부(117)의 제4다중화기(119a)는 상기 버스(200)를 통해 들어온 모드 신호의 값에 따라서 출력이 제어되어 상기 제4XOR연산기(118b)의 입력신호를 선택적으로 결정한다.

<69> 상기 라운드 XOR연산부(117)의 제5XOR연산기(118c)는 새로운 라운드의 128비트 라운드 키 중 63번째부터 32번째까지의 32비트 라운드 키(RK2)와 이전 라운드의 128비트 라운드 키 중 31번째부터 0번째까지의 32비트 라운드 키(PK3)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 31번째부터 0번째까지의 32비트 라운드 키(RK3)를 생성한다.

<70> 상기 제5XOR연산기(118c)는 이전 라운드의 128비트 라운드 키의 63번째부터 32번째까지의 32비트 라운드 키(PK2)와 그 다음 31번째부터 0번째까지의 32비트 라운드 키(PK3)를 XOR연산하여 새로운 라운드의 복호화용 128비트 라운드 키 중 31번째부터 0번째까지의 32비트 라운드 키(RK3)를 생성한다.

<71> 상기 라운드 XOR연산부(117)의 제5다중화기(119b)는 상기 버스(200)를 통해 들어온 모드 신호의 값에 따라서 출력이 제어되어 상기 제5XOR연산기(118c)의 입력신호를 선택적으로 결정한다.

<72> 상기와 같이 구성되는 본 발명에 따른 라인달 블록 암복호 장치는 다음과 같이 작동하여 암호화 및 복호화 과정을 수행한다.

<73> 먼저, 도 1과 도 2를 참조하여 라인달 블록 암호 장치의 암호화 및 복호화 동작을 살펴보면 다음과 같다.

74> 라운드 연산이 시작되면 최초의 128비트 입력 키(key)가 버스(200)를 통해서 라운드 키 생성부(100)로 들어감에 따라서 라운드 키 생성 과정이 진행되고, 128비트 입력 데이터(data)는 시프트/역시프트_로우 변환부(120)에 들어간다. 여기서, 상기 128비트 입력데이터는 버스(200)로부터 128비트 데이터 레지스터(500)를 경유하여 상기 시프트/역시프트_로우 변환부(120)로 들어온 것이다(도1 및 도2 참조). 이와 같은 방법으로 상기 입력데이터가 버스(200)로부터 라운드연산부(100)(즉, 시프트/역시프트_로우 변환부(120))로 제공되도록 하는 구성은 상기 128비트 데이터 레지스터(500)가 입력데이터 레지스터의 기능도 동시에 수행할 수 있도록 할 수 있다는 이점이 있다. 그러나 설계자의 의도에 따라 이 부분의 구성은 얼마든지 달리 할 수 있음은 물론이다.

75> 이렇게 입력된 상기 128비트 입력 데이터에 대해, 상기 시프트/역시프트_로우 변환부(120)는 암호/복호 과정에 따라 라인달 블록 암호알고리즘에 정의된 대로 각기 다른 수만큼의 바이트 시프트/역시프트 변환을 수행한다.

76> 만약, 상기 라운드 연산 제어부(300)에서 상위 64비트를 선택하는 신호(sel='1')를 보내오면 상기 시프트/역시프트_로우 변환부(120)는 제1다중화기(121)를 통해서 상위 64비트를 출력하고, 상기 라운드 연산 제어부(300)에서 하위 64비트를 선택하는 신호(sel='0')를 보내오면 제1다중화기(121)를 통해서 하위 64비트를 출력한다.

77> 상기와 같이 바이트 시프트/역시프트_로우 변환이 수행되고 나면, 다음으로 상기 상위 또는 하위 64비트 데이터는 대치/역대치 연산을 수행하는 대치/역대치 변환부(130)로 들어가서 대치 박스나 역대치 박스에 의해 대치 또는 역대치 연산된다. 이때, 상기 대치 박스와 역대치 박스는 라인달 알고리즘의 스펙에 정의된 바와 같이 한 바이트의 입력에 대해서 한 바이트의 출력을 내보내는 대치 변환기로써 작용한다. 또한, 본 발명에서 제안하는 대치/역대치 변환부

(130)는 한번에 64비트의 데이터만 처리하면 되기 때문에 8개의 대치 박스나 8개의 역대치 박스만이 필요하다.

78> 상기와 같이 대치/역대치 변환이 수행되고 난 상태에서, 상기 버스(200)에서 암호 과정을 선택하는 모드신호(mode='0')를 보내오면 상기 상위 또는 하위 64비트 데이터는 제1역다중화기(140)의 암호 출력단(0)을 통해서 믹스/역믹스컬럼 변환부(150)의 입력으로 들어가고, 상기 버스(200)에서 복호 과정을 선택하는 모드신호(mode='1')를 보내오면 상기 상위 또는 하위 64비트 데이터는 제1역다중화기(140)의 복호 출력단(1)을 통해서 라운드 키 더하기 변환부(170)의 입력으로 들어간다.

79> 믹스/역믹스컬럼 변환을 거친 64비트 데이터는 상기 버스(200)에서 암호 과정을 선택하는 모드신호(mode='0')를 보내오면 제2역다중화기(160)의 암호 출력단(0)을 통해서 라운드 키 더하기 변환부(170)의 입력으로 들어가고, 상기 버스(200)에서 복호 과정을 선택하는 모드신호(mode='1')를 보내오면 상기 제2역다중화기(160)의 복호 출력단(1)을 통해서 라운드 연산의 결과 데이터로 출력된다.

80> 또한, 상기 라운드 키 더하기 변환을 거친 64비트 데이터는 상기 버스(200)에서 암호 과정을 선택하는 모드신호(mode='0')를 보내오면 제3역다중화기(180)의 암호 출력단(0)을 거쳐 라운드 연산의 출력 결과가 되고, 상기 버스(200)에서 복호 과정을 선택하는 모드신호(mode='1')를 보내오면 제3역다중화기(180)의 복호 출력단(1)을 거쳐 믹스/역믹스컬럼 변환부(150)의 입력이 된다.

81> 이상의 과정은 상위 64비트 및 하위 64비트 데이터 각각에 대해 모두 수행되며(처리 타이밍에 대해서는 후술함), 이것이 하나의 라운드가 되며, 이러한 과정이 소정의 라운드만큼 반복된다.

2> 상기한 바와 같이, 본 발명에서는 암호화 과정과 복호화 과정에서 공통으로 사용되는 구성요소를 공유하여 하드웨어 리소스의 사용을 줄이고자 하였기 때문에 각 변환부는 암호화 변환과 복호화 변환의 기능을 모두 갖고 있다.

3> 한편, 도 3을 참조하여 라운드 키 생성부(110)가 본 발명에 따른 라인달 블록 암호 장치의 암호화 및 복호화 동작에 필요한 암호화용 또는 복호화용 라운드 키를 생성하는 동작을 살펴보면 다음과 같다.

4> 상기 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면 이때부터 라운드 연산이 시작된다.

5> 라운드 연산이 시작되면 상기 라운드 키 생성부(110)는 128비트 프리 키 레지스터(111)에 저장된 이전 라운드의 128비트 라운드 키(prekey)를 이용하여 새로운 라운드의 라운드 키(RK)를 생성하기 시작한다.

6> 상기 버스(200)에서 암호 과정을 선택하는 모드신호(mode='0')를 보내오는 경우, 먼저 상기 128비트 프리 키 레지스터(111)의 이전 라운드의 128비트 라운드 키 중 최하위 32비트(PK3)가 제2다중화기(113)를 거쳐 시프트기(114)의 입력으로 들어간다.

7> 반면에, 상기 버스(200)에서 복호 과정을 선택하는 모드신호(mode='1')를 보내오는 경우, 먼저 상기 제5XOR연산기(118c)가 이전 라운드의 라운드 키 중 하위 64비트(PK2,PK3)를 32비트씩 서로 XOR연산하여 새로운 라운드 키의 최하위 32비트(RK3)로 임시 저장함과 동시에 이 값(RK3)은 상기 제2다중화기(113)를 거쳐 시프트기(114)의 입력으로 들어간다.

8> 상기 시프트기(114)로 들어간 32비트 키는 한 바이트 왼쪽으로 순환 시프트 연산된 다음, 4개의 대치 박스로 이루어진 대치변환기(115)로 들어가 대치 변환된다.

> 상기와 같이 대치 변환을 마친 32비트 키들 중 최상위 8비트 키는 상기 제1XOR연산기(116)에 의해 상기 라운드 연산 제어부(300)로부터 들어오는 라운드 번호신호가 나타내는 라운드가 몇 번째 라운드인지에 따라서 결정되는 상수값(Rcon)과 XOR연산되고, 이 제1XOR연산기(116)의 결과값 8비트와 상기 대치변환기(115)의 대치 연산 결과인 나머지 24비트가 합해져서 상기 라운드 XOR연산부(117)의 제2XOR연산기(118)의 입력으로 들어간다.

>> 특히, 이때 라운드 키 생성 과정 중 라운드 번호와 관계되는 상수값을 XOR연산을 하는 부분을 상기 대치변환기(115)를 통과한 32비트 데이터 중 상위 8비트만으로 한정함으로써 하드웨어 면적 감소 효과를 얻을 수 있다. 한편, 이 부분에 대해서 라인달 알고리즘 스펙에서는 라운드 번호와 관계되는 상수값을 8비트의 상수값에 24비트의 '0'을 패딩하여 32비트를 만든 후 대치변환기(115)를 통과한 32비트 값과 XOR연산하는 구조로 설명되어 있다.

>11> 이어서, 상기 제2XOR연산기(118)는 제1XOR연산기(116)의 결과값 8비트와 상기 대치변환기(115)의 대치 연산 결과인 나머지 24비트가 합해진 값과 이전 라운드의 라운드 키 중 최상위 32비트(PK0)를 XOR연산하여 생성한 결과값을 새로운 라운드의 최상위 32비트 라운드 키(RK0)로 저장한다.

>92 상기와 같이 새로운 라운드의 암호화 또는 복호화에 필요한 최상위 32비트 라운드 키(RK0)가 생성되고 나면, 다음으로 상기 제3XOR연산기(118a)는 암호화 과정의 경우 새로운 라운드의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 상위 95번째부터 64번째까지의 32비트 라운드 키(PK1)를 XOR연산을 수행하여 새로운 라운드의 다음 32비트 라운드 키(RK1)를 생성하고, 복호화 과정의 경우 이전 라운드의 최상위 32비트(PK0)와 이전 라운드의 다음 상위 32비트(PK1)를 XOR연산하여 새로운 라운드의 다음 32비트 라운드 키(RK1)를 생성한다.

이때, 상기 제3다중화기(119)는 버스(200)를 통해 들어오는 암호 과정 혹은 복호 과정을 나타내는 모드신호에 따라서 제3XOR연산기(118a)의 입력 값을 결정한다.

상기와 같이 새로운 라운드의 최상위 32비트 라운드 키(RK0)의 다음 32비트 라운드 키(RK1)가 생성되고 나면, 그 다음의 암호화용 또는 복호화용 32비트 라운드 키(RK2)와 최하위 32비트 라운드 키(RK3)는 각각 상기 제3XOR연산기(118a)와 동일한 방식으로 작동하는 상기 제4XOR연산기(118b)와 제5XOR연산기(118c)에 의해 생성되고, 상기 제4다중화기(119a)는 제4XOR연산기(118b)의 입력 값을 결정하며, 상기 제5다중화기(119b)는 제5XOR연산기(118c)의 입력 값을 결정한다.

특히, 이와 같이 최상위 비트부터 최하위 비트까지 32비트씩 나누어 새로운 라운드의 128비트 라운드 키를 생성하는데 소요되는 시간은 암호 과정의 경우 상기 라운드 연산 제어부(300)로부터 들어오는 라운드 연산 시작신호의 총 4클럭 기간에 해당하고, 복호 과정의 경우 총 2클럭 기간에 해당한다.

실제로, 암호화 라운드 연산 시작 신호의 1번째 클럭이 1이 되면 상기 제2XOR연산기(118)에 의해 새로운 라운드의 최상위 32비트 라운드 키(RK0)가 생성되고, 2번째, 3번째, 4번째 클럭이 1이 될 때마다 상기 제3XOR연산기(118a)와 제4XOR연산기(118b) 및 제5XOR연산기(118c)에 의해 새로운 라운드의 32비트 라운드 키 RK1, RK2, RK3이 생성된다. 또한, 복호화 라운드 연산 시작 신호의 1번째 클럭이 1이 되면 상기 제2XOR연산기(118)에 의해 새로운 라운드의 최상위 32비트 라운드 키(RK0)가 생성되고, 2번째 클럭이 1이 될 때 상기 제3XOR연산기(118a)와 제4XOR연산기(118b) 및 제5XOR연산기(118c)에 의해 새로운 라운드의 32비트 라운드 키 RK1, RK2, RK3이 동시에 생성된다.

> 상기 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호가 라운드 연산부(100)로 들어오는 경우, 상기 라운드 키 생성부(110)는 2클럭 기간에 암호화 라운드 키를 생성한다.

3> 이때, 상기 제2XOR연산기(118)가 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)를 생성하기까지의 과정은 라운드 연산 시작신호의 1번째 클럭이 '1'이 되면 진행된다.

9> 이어서, 라운드 연산 시작신호의 2번째 클럭이 '1'이 되면 상기 제3XOR연산기(118a)는 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 다음 32비트 라운드 키(PK1)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 95번째부터 64번째까지의 32비트 라운드 키(RK1)를 생성한다.

00> 이와 동시에, 상기 제4XOR연산기(118b)는 제3XOR연산기(118a)가 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 32비트 다음의 95번째부터 64번째까지의 32비트 라운드 키(PK1)를 XOR연산한 결과값(RK0⊕PK1)과 이전 라운드의 63번째부터 32번째까지의 32비트 라운드 키(PK2)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 63번째부터 32번째까지의 32비트 라운드 키(RK2)를 생성한다.

101> 이와 동시에, 상기 제5XOR연산기(118c)는 제4XOR연산기(118b)가 제3XOR연산기(118a)에 의해 XOR연산된 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 32비트 다음의 95번째부터 64번째까지의 32비트 라운드 키(PK1)의 XOR연산 결과값(RK0⊕PK1)과 이전 라운드의 63번째부터

32번째까지의 32비트 라운드 키(PK2)를 XOR연산한 결과값(RK0⊕PK1⊕PK2)과 이전 라운드의 31번째부터 0번째까지의 32비트 라운드 키(PK3)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 31번째부터 0번째까지의 32비트 라운드 키(RK3)를 생성한다.

2> 상기 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호가 라운드 연산부(100)로 들어오는 경우, 상기 라운드 키 생성부(110)는 1클럭 기간에 암호화 라운드 키를 생성한다.

)3> 이때, 상기 제2XOR연산기(118)가 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)를 생성하기까지의 과정은 라운드 연산 시작신호가 입력됨과 동시에 클럭이 '0'인 상태에서 진행된다.

)4> 상기 라운드 연산 시작신호의 1번째 클럭이 '1'이 되면 상기 제3XOR연산기(118a)는 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 다음 32비트 라운드 키(PK1)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 95번째부터 64번째까지의 32비트 라운드 키(RK1)를 생성한다.

.05> 이와 동시에, 상기 제4XOR연산기(118b)는 제3XOR연산기(118a)가 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 32비트 다음의 95번째부터 64번째까지의 32비트 라운드 키(PK1)를 XOR연산한 결과값(RK0⊕PK1)과 이전 라운드의 63번째부터 32번째까지의 32비트 라운드 키(PK2)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 63번째부터 32번째까지의 32비트 라운드 키(RK2)를 생성한다.

>> 이와 동시에, 상기 제5XOR연산기(118c)는 제4XOR연산기(118b)가 제3XOR연산기(118a)에 의해 XOR연산된 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)와 이전 라운드의 128비트 라운드 키 중 최상위 32비트 다음의 95번째부터 64번째까지의 32비트 라운드 키(PK1)의 XOR연산 결과값(RK0⊕PK1)과 이전 라운드의 63번째부터 32번째까지의 32비트 라운드 키(PK2)를 XOR연산한 결과값(RK0⊕PK1⊕PK2)과 이전 라운드의 31번째부터 0번째까지의 32비트 라운드 키(PK3)를 XOR연산하여 새로운 라운드의 암호화용 128비트 라운드 키 중 31번째부터 0번째까지의 32비트 라운드 키(RK3)를 생성한다.

)7> 상기 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호가 라운드 연산부(100)로 들어오는 경우, 상기 라운드 키 생성부(110)는 1클럭 기간에 복호화 라운드 키를 생성한다.

08> 이때, 상기 제2XOR연산기(118)가 새로운 라운드의 128비트 라운드 키 중 127번째부터 96번째까지의 최상위 32비트 라운드 키(RK0)를 생성하기까지의 과정은 라운드 연산 시작신호가 입력됨과 동시에 클럭이 '0'인 상태에서 진행된다.

.09> 상기 라운드 연산 시작신호의 1번째 클럭이 '1'이 되면 상기 제3XOR연산기(118a)는 이전 라운드의 최상위 32비트(PK0)와 이전 라운드의 다음 상위 32비트(PK1)를 XOR연산하여 새로운 라운드의 다음 32비트 라운드 키(RK1)를 생성하고, 연속해서 그 다음의 복호화용 32비트 라운드 키(RK2)와 최하위 32비트 라운드 키(RK3)는 각각 상기 제3XOR연산기(118a)와 동일한 방식으로 작동하는 상기 제4XOR연산기(118b)와 제5XOR연산기(118c)에 의해 생성된다. 이들 과정은 1번째 클럭 기간에서 동시에 수행된다.

> 상기와 같이 작동하여 암호화 및 복호화 과정을 수행하는 라인달 블록 암호 장치의 작동을 상기 라운드 연산 제어부(300)로부터 라운드 연산부(100)로 들어가는 라운드 연산 시작신호의 클럭 수에 따라 구분하여 보다 구체적으로 설명하면 다음과 같다.

> 도 4는 본 발명에 따른 라인달 블록 암호의 암호화 방법을 도시한 제1타이밍도이다.

> 도 4를 참조하면, 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면(S400), 라운드 연산 시작신호의 1번째 클럭이 '1'이 되는 순간에 128비트 라운드 연산 입력 데이터 중 상위 64비트 데이터의 바이트 시프트 변환과 대치 연산이 순차적으로 수행되는데(S401), 이들 두 과정이 한 클럭 안에 이루어진다. 이들 과정의 수행 결과는 상기 64비트 데이터 레지스터(400)에 저장된다. 또한, 상기 라운드 연산 시작신호가 들어오면 라운드 연산 시작신호의 1번째 클럭이 '1'이 되는 순간에 128비트 라운드 입력 키를 이용하여 128비트 라운드 키 생성 과정이 시작된다(S401a).

> 라운드 연산 시작신호의 2번째 클럭이 '1'이 되는 순간 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터를 이용하여 믹스컬럼 변환이 시작되고 결과값은 64비트 데이터 레지스터(400)에 저장되며(S402), 동시에 라운드 연산 입력 데이터의 하위 64비트 데이터의 바이트 시프트 변환과 대치 연산이 순차적으로 수행되는데(S402), 이들 과정도 한 클럭 안에 수행된다. 또한, 상기 하위 64비트 데이터의 바이트 시프트 변환과 대치 연산의 결과 데이터는 라운드 연산 결과를 저장하는 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다.

> 라운드 연산 시작신호의 3번째 클럭이 '1'이 되는 순간 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터는 라운드 키 생성부(110)에서 생성된 라운드 키의 상위 64비트와 더해지기 위해서 라운드 키 더하기 변환부(170)의 입력으로 들어가고 그 결과값은 상기 128비트 데이터 레지스터(500)의 상위 64비트 위치에 저장되며(S403), 상기 128비트 데이터 레지

스터(500)의 하위 64비트 데이터는 믹스컬럼 변환과정을 거치고 그 결과값은 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S403).

<3> 라운드 연산 시작신호의 4번째 클럭이 '1'이 되면 상기 128비트 데이터 레지스터(500)의 하위 64비트 데이터는 라운드 키 생성부(110)에서 생성된 라운드 키의 하위 64비트와 더해지기 위해서 라운드 키 더하기 변환부(170)의 입력으로 들어가고 그 결과값은 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S404).

<6> 따라서, 상기와 같이 암호화 과정을 수행하는 라인달 블록 암호 장치에서는 상기 128비트 데이터 레지스터(500)의 128비트 데이터가 다음 라운드의 128비트 라운드 연산 입력 데이터로 되고, 라운드 키 생성부(110)에서 새롭게 생성되어 상기 128비트 라운드 키 레지스터(111a)에 저장된 라운드 키(RK) 또한 128비트 프리 키 레지스터(111)에 저장되었다가 다음 라운드의 128비트 라운드 입력 키로 사용된다. 이로써, 한 라운드의 암호화 연산이 완료되며 총 4클럭이 소요된다.

<17> 도 4에 도시된 암호화 방법을 본 발명에 따른 라인달 블록 암호 장치에서 수행하는 경우, 상기 라운드 키 생성부(110)는 라운드 키 생성 과정을 라운드 연산 시작신호의 총 4클럭 기간에 끝낸다. 즉, 도 4에 따르면 상위 64비트 데이터와 라운드 키를 더하는 과정인 라운드 키 더하기 변환 과정(S403)이 라운드 연산을 시작한 후 3번째 클럭 후에 이루어진다. 라운드 연산을 시작 한 후 2번째 클럭이 지나면 새로운 라운드 키 중 상위 64비트 라운드 키만이 생성된 상태이고, 이 시점에서 상위 64비트 라운드 키만을 사용하므로 라운드 연산의 암호화 동작에는 무리가 없다. 또한 라운드 연산을 시작한 후 3번째 클럭이 지나고 4번째 클럭이 시작되는 시점에서는 128비트 라운드 키가 모두 생성되는 시점과 일치하므로 하위 64비트 데이터와 하위 64비트 라운드 키를 더하는 라운드 키 더하기 변환 과정(S404)의 수행에도 문제가 없다.

▷ 또한, 상기와 같이 암호화 과정을 수행하는 라인달 블록 암호 장치에서는 64비트 데이터 레지스터(400)를 암호화 과정 중에 생성되는 중간 데이터 저장공간으로 사용함으로써 상위 64비트 데이터의 바이트 시프트 변환 결과가 하위 64비트 바이트 시프트 변환에 영향을 주지 않으며, 상위 64비트 데이터와 하위 64비트 데이터의 변환을 동시에 수행하되 동일한 클럭 주기 동안에 상위 64비트 데이터와 하위 64비트 데이터가 동일한 변환을 수행하지 않기 때문에 변환에 필요한 하드웨어 모듈의 수를 절반으로 줄일 수 있다.. 특히, 각 클럭마다 생성된 데이터를 하나의 저장공간에 계속 업데이트 하는 방식을 취함으로써 추가적인 저장공간을 사용할 필요가 없다. 즉, 이 경우는 파이프 라인 구조를 응용하되 추가의 하드웨어가 필요하지 않는 구조를 지향하였으며 이들 구조는 이하에서 다른 실시예로 설명할 라인달 블록 암호의 암호화 및 복호화 방법들에서도 동일하게 적용된다.

19> 도 5는 본 발명에 따른 라인달 블록 암호의 복호화 방법을 도시한 제1타이밍도이다.

20> 도 5를 참조하면, 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면(S500), 라운드 연산 시작신호의 1번째 클럭이 '1'이 되는 순간에 128비트 라운드 연산 입력 데이터 중 상위 64비트 데이터의 바이트 역시프트 변환과 역대치 연산이 순차적으로 수행되는데(S501), 이들 두 과정이 한 클럭 안에 이루어진다. 이때, 결과 데이터는 상기 64비트 데이터 레지스터(400)에 저장된다. 또한, 라운드 연산 시작신호가 들어오면 라운드 연산 시작신호의 1번째 클럭이 '1'이 되는 순간에 128비트 라운드 입력 키를 이용하여 128비트 라운드 키 생성 과정이 시작된다(S501a).

121> 라운드 연산 시작신호의 2번째 클럭이 '1'이 되는 순간 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터와 라운드 키 생성부(110)를 통해서 생성된 라운드 키의 상위 64비트가 더해지는 라운드 키 더하기 변환이 시작되고 결과 데이터는 64비트 데이터 레지스터

(400)에 저장되며(S502), 동시에 라운드 입력 데이터의 하위 64비트 데이터의 바이트 역시프트 변환과 역대치 연산이 순차적으로 수행되고 결과 데이터는 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S502).

2> 라운드 연산 시작신호의 3번째 클럭이 '1'이 되는 순간 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터는 믹스/역믹스컬럼 변환부(150)의 입력으로 들어가고 역믹스컬럼 변환의 결과 데이터는 상기 128비트 데이터 레지스터(500)의 상위 64비트 위치에 저장되며(S503), 동시에 역대치 연산을 마친 상기 하위 64비트 데이터는 라운드 키 생성부(110)에서 생성된 라운드 키와 더해지는 라운드 키 더하기 변환을 거치고 결과 데이터는 상기 128비트 데이터 레지스터의 하위 64비트 위치에 저장된다(S503).

23> 라운드 연산 시작신호의 4번째 클럭이 '1'이 되면 라운드 키 더하기 변환을 마친 상기 하위 64비트 데이터는 믹스/역믹스컬럼 변환부(150)의 입력이 되어 역믹스컬럼 변환되고 결과 데이터는 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S504).

24> 이때도, 상기 128비트 데이터 레지스터(500)의 128비트 데이터는 다음 복호화 라운드 연산의 128비트 라운드 연산 입력 데이터로 사용되고, 라운드 키 생성 결과인 128비트 라운드 키(RK)는 상기 128비트 프리 키 레지스터(111)에 저장되어 다음 라운드 연산의 128비트 라운드 입력 키로 사용된다. 이로써, 한 라운드의 복호화 연산이 완료되며 총 4클럭이 소요된다.

25> 도 5에 도시된 복호화 방법을 본 발명에 따른 라인달 블록 암호 장치에서 수행하는 경우, 상기 라운드 키 생성부(110)는 라운드 키 생성 과정을 라운드 연산 시작신호의 총 2클럭 기간에 끝낸다. 즉, 도 5에 따르면 상위 64비트 라운드 키와 상위 64비트 데이터를 더하는 라운드 키 더하기 변환 과정(S502)이 라운드 연산을 시작 한 후 2번째 클럭에서 이루어지므로, 2번

째 클럭에서는 이미 128비트 라운드 키가 모두 생성된 상태로 되어 라운드 연산의 수행에 문제가 없다.

:6> 도 6은 본 발명에 따른 라인달 블록 암호의 암호화 방법을 도시한 제2타이밍도이다.

:7> 도 6을 참조하면, 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면(S600), 라운드 연산 시작신호의 1번째 클럭이 '1'이 되는 순간 상위 64비트 데이터의 바이트 시프트 연산과 대치 연산이 순서대로 수행되어 결과 데이터는 64비트 데이터 레지스터(400)에 저장된다(S601). 또한, 라운드 키 생성 과정이 이와 동시에 수행된다(S601a).

28> 라운드 연산 시작신호의 2번째 클럭이 '1'이 되면 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터가 믹스컬럼 변환을 한 후 라운드 키 생성부(110)의 결과 데이터 중 상위 64비트 라운드 키와 라운드 키 더하기 변환되고 결과 데이터는 상기 64비트 데이터 레지스터(400)에 저장되며(S602), 이와 동시에 하위 64비트 데이터의 바이트 시프트 변환과 대치 변환이 차례대로 수행되어 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S602).

129> 라운드 연산 시작신호의 3번째 클럭이 '1'이 되면 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터가 상기 128비트 데이터 레지스터(500)의 상위 64비트 위치에 저장되며, 상기 128비트 데이터 레지스터(500)의 하위 64비트 데이터는 믹스컬럼 변환을 한 다음 라운드 키 생성부(110)에서 생성된 라운드 키 중 하위 64비트 라운드 키와 라운드 키 더하기 변환을 수행하고 결과 데이터는 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S603).

0> 이때도, 상기 128비트 데이터 레지스터(500)의 128비트 데이터는 다음 라운드 연산의 128비트 라운드 연산 입력 데이터로 사용되고, 라운드 키 생성부(110)에서 생성된 라운드 키(RK)는 128비트 프리 키 레지스터(111)에 저장되어 다음 라운드의 128비트 라운드 입력 키로 사용된다. 이로써, 한 라운드의 암호화 연산이 완료되며 총 3클럭이 소요된다.

1> 도 6에 도시된 암호화 방법을 본 발명에 따른 라인달 블록 암호 장치에서 수행하는 경우, 상기 라운드 키 생성부(110)는 라운드 키 생성 과정을 라운드 연산 시작신호의 총 2클럭 기간에 끝낸다. 즉, 도 6에 따르면 상위 64비트 라운드 키와 상위 64비트 데이터를 더하는 라운드 키 더하기 변환 과정(S602)이 라운드 연산을 시작 한 후 2번째 클럭에서 이루어지므로, 2번째 클럭에서는 이미 128비트 라운드 키가 모두 생성된 상태로 되어 라운드 연산의 수행에 문제가 없다.

2> 도 7은 본 발명에 따른 라인달 블록 암호의 복호화 방법을 도시한 제2타이밍도이다.

3> 도 7을 참조하면, 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면(S700), 라운드 연산 시작신호의 1번째 클럭이 '1'이 되면 라운드 입력 데이터의 상위 64비트 데이터가 바이트 역시프트 변환과 역대치 변환을 거치고 이 결과 데이터는 64비트 데이터 레지스터(400)에 저장된다(S701). 또한, 이들 변환과 동시에 라운드 키 생성 과정이 시작된다(S701a).

4> 라운드 연산 시작신호의 2번째 클럭이 '1'이 되면 상기 64비트 데이터 레지스터(400)의 64비트 데이터와 라운드 키 생성부(110)에서 생성된 라운드 키의 상위 64비트 라운드 키가 라운드 키 더하기 변환되고 이 결과 데이터가 믹스/역믹스컬럼 변환부(150)의 입력 데이터로 되고, 역믹스컬럼 변환된 결과 데이터가 상기 64비트 데이터 레지스터(400)에 저장된다(S702). 이들 라운드 키 더하기 변환 및 역믹스컬럼 변환과 동시에 라운드 입력 데이터의 하위 64비트

데이터의 바이트 역시프트 변환과 역대치 변환이 차례로 수행되고 결과 데이터가 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S702).

5> 라운드 연산 시작신호의 3번째 클럭이 '1'이 되면 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터가 상기 128비트 데이터 레지스터(500)의 상위 64비트 위치에 저장되며, 상기 128비트 데이터 레지스터(500)의 하위 64비트 데이터와 라운드 키 생성부(110)의 라운드 키 중 하위 64비트 라운드 키가 라운드 키 더하기 변환되고 이 결과 데이터를 역믹스컬럼 변환하여 얻은 결과 데이터가 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S703).

36> 이때도, 상기 128비트 데이터 레지스터(500)의 128비트 데이터는 다음 라운드 연산의 128비트 라운드 입력 데이터로 이용되고, 라운드 키 생성부(110)에서 생성된 라운드 키(RK)가 상기 128비트 프리 키 레지스터(111)에 저장되어 다음 라운드 연산의 128비트 라운드 입력 키로 사용된다. 이로써, 한 라운드의 복호화 연산이 완료되며 총 3클럭이 소요된다.

37> 도 7에 도시된 복호화 방법을 본 발명에 따른 라인달 블록 암호 장치에서 수행하는 경우, 상기 라운드 키 생성부(110)는 라운드 키 생성 과정을 라운드 연산 시작신호의 총 2클럭 기간에 끝낸다. 즉, 도 7에 따르면 상위 64비트 라운드 키와 상위 64비트 데이터를 더하는 라운드 키 더하기 변환 과정(S702)이 라운드 연산을 시작 한 후 2번째 클럭에서 이루어지므로, 2번째 클럭에서는 이미 128비트 라운드 키가 모두 생성된 상태로 되어 라운드 연산의 수행에 문제가 없다.

138> 도 8은 본 발명에 따른 라인달 블록 암호의 암호화 방법을 도시한 제3타이밍도이다.

▷ 도 8을 참조하면, 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작 신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면(S800), 라운드 연산 시작신호의 1번째 클럭이 '1'이 되면 라운드 입력 데이터의 상위 64비트 데이터에 대한 바이트 시프트 변환, 대치변환, 믹스컬럼 변환, 라운드 키 더하기 변환이 차례로 수행되고 결과 데이터는 상기 64비트 데이터 레지스터(400)에 저장된다(S801). 이와 동시에 라운드 키 생성 과정(S801a)이 진행되고 생성된 라운드 키 중 상위 64비트 라운드 키가 앞서 기술한 라운드 키 더하기 변환에 사용된다. 이들 과정은 한 클럭 내에 수행된다.

.0> 라운드 연산 시작신호의 2번째 클럭이 '1'이 되면 라운드 입력 데이터의 하위 64비트 데이터에 대한 바이트 시프트 변환, 대치 변환, 믹스컬럼 변환, 라운드 키 더하기 변환이 차례로 수행되고 결과 데이터는 상기 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S802). 그리고 라운드 키 생성 과정에서 생성된 라운드 키 중 하위 64비트 라운드 키가 앞서 기술한 라운드 키 더하기 변환에 사용된다. 이때, 64비트 데이터 레지스터(400)에 저장된 64비트 데이터는 128비트 데이터 레지스터(500)의 상위 64비트 위치에 저장되고, 라운드 키 생성부(110)에서 새롭게 생성된 128비트 라운드 키(RK)는 128비트 라운드 키 레지스터(111a)에 저장되고 상기 128비트 프리 키 레지스터(111)에 백업된다. 이로써, 한 라운드의 암호화 연산이 완료되며 총 2클럭이 소요된다.

.41> 도 8에 도시된 암호화 방법을 본 발명에 따른 라인달 블록 암호 장치에서 수행하는 경우, 상기 라운드 키 생성부(110)는 라운드 키 생성 과정을 라운드 연산 시작신호의 총 1클럭 기간에 끝낸다. 즉, 도 8에 따르면 상위 64비트 라운드 키와 상위 64비트 데이터를 더하는 라운드 키 더하기 변환 과정(S801)이 라운드 연산을 시작 한 후 1번째 클럭에서 이루어지므로, 1

번째 클럭에서는 이미 128비트 라운드 키가 모두 생성된 상태로 되어 라운드 연산의 수행에 문제가 없다.

[42] 실제로, 이때는 도 3에 도시된 라운드 키 생성부(110)가 RK0을 이용하여 RK1을 생성하고, RK1을 이용하여 RK2를 생성하고, RK2를 이용하여 RK3을 생성하지 않고, RK0을 라운드 연산 시작신호가 입력됨과 동시에 클럭이 '0'인 상태에서 생성해두고, 라운드 연산 시작신호의 1번째 클럭이 '1'이 되는 순간 RK0과 PK1을 XOR연산해서 RK1을 생성하고, 이와 동시에 RK0과 PK1, PK2를 XOR연산해서 RK2를 생성하고, 또 이와 동시에 RK0과 PK1, PK2, PK3을 XOR해서 RK3을 생성한다.

[43] 도 9는 본 발명에 따른 라인달 블록 암호의 복호화 방법을 도시한 제3타이밍도이다.

[44] 도 9를 참조하면, 라운드 연산 제어부(300)로부터 클럭 주기가 일정한 라운드 연산 시작신호와 라운드 번호신호가 라운드 연산부(100)로 들어오면(S900), 라운드 연산 시작신호의 1번째 클럭이 '1'이 되면 라운드 입력 데이터의 상위 64비트 데이터에 대한 바이트 역시프트 변환, 역대치 변환, 라운드 키 더하기 변환, 역믹스컬럼 변환이 차례로 수행되고 결과 데이터는 64비트 데이터 레지스터(400)에 저장된다(S901). 이들 과정은 한 클럭에 수행된다. 또한, 이와 동시에 복호화용 라운드 키 생성 과정이 진행(S901a)되어 라운드 키 생성부(110)에서 생성된 라운드 키 중 상위 64비트 라운드 키가 앞서 기술한 라운드 키 더하기 변환 과정에 사용된다.

[45] 라운드 연산 시작신호의 2번째 클럭이 '1'이 되면 하위 64비트 데이터에 대한 바이트 역시프트 변환, 역대치 변환, 라운드 키 더하기 변환, 역믹스컬럼 변환이 차례로 수행되고 결과 데이터는 128비트 데이터 레지스터(500)의 하위 64비트 위치에 저장된다(S902). 이들 과정은 한 클럭에 수행된다. 그리고, 상기 라운드 키 더하기 변환에는 라운드 키 생성부(110)에서 한

클럭 앞서 생성된 라운드 키 중 하위 64비트 라운드 키가 사용된다. 이때, 상기 64비트 데이터 레지스터(400)에 저장된 64비트 데이터는 128비트 데이터 레지스터(500)의 상위 64비트 위치에 저장되며, 라운드 키 생성부(110)에서 새로 생성된 128비트 라운드 키(RK)는 128비트 라운드 키 레지스터(111a)에 저장되고 상기 128비트 프리 키 레지스터(111)에 백업된다. 이로써, 한 라운드의 복호화 연산이 완료되며 총 2클럭이 소요된다.

6> 도 9에 도시된 복호화 방법을 본 발명에 따른 라인달 블록 암호 장치에서 수행하는 경우, 상기 라운드 키 생성부(110)는 라운드 키 생성 과정을 라운드 연산 시작신호의 총 1클럭 기간에 끝낸다. 즉, 도 9에 따르면 상위 64비트 라운드 키와 상위 64비트 데이터를 더하는 라운드 키 더하기 변환 과정(S901)이 라운드 연산을 시작 한 후 라운드 연산 시작신호의 1번째 클럭에서 이루어지나 1번째 클럭에서는 이미 128비트 라운드 키가 모두 생성된 상태로 되어 라운드 연산의 수행에 문제가 없다.

47> 실제로, 이때는 도 3에 도시된 라운드 키 생성부(110)가 RK0을 라운드 연산 시작신호가 입력됨과 동시에 클럭이 '0'인 상태에서 생성해두고, 1번째 클럭이 '1'이 되는 순간 PK0과 PK1을 XOR연산해서 RK1을 생성하고, 이와 동시에 PK1과 PK2를 XOR연산해서 RK2를 생성하고, 또 이와 동시에 PK2와 PK3을 XOR해서 RK3을 생성한다.

48> 상기한 바와 같은 도 8에 도시된 암호화 방법과 도 9에 도시된 복호화 방법에 따른 라인달 블록 암호 장치는 적은 면적과 적은 전력 소모 및 낮은 동작 주파수 특성을 가지는 스마트 카드, USIM(user subscriber identity module) 카드, SIM 카드 등에 적용하기에 적당한 모델이다.

【발명의 효과】

> 상술한 바와 같이 본 발명에 따른 라인달 블록 암호 장치와 그 암호화 및 복호화 방법은, 고속, 저면적의 암호 프로세서를 요구하는 휴대폰이나 PDA같은 이동단말기 또는 스마트 카드에 탑재함으로써 보안이 필요한 중요한 데이터를 빠른 시간 안에 암호화 및 복호화할 수 있고, 예컨대 128비트 입력 데이터를 상위 64비트와 하위 64비트로 나누어 라운드 연산을 수행하도록 되어 있기 때문에, 다음과 같은 효과를 가진다.

)> 첫째, 본 발명에 따른 암호 장치 내의 라운드 연산 장치를 반복 사용함으로써 적은 면적과 빠른 속도로 실시간 데이터를 암호화 및 복호화할 수 있다.

1> 둘째, 본 발명에 따른 암호 장치는 라인달 알고리즘을 적용한 라운드 연산 장치를 이용하여 블록 암호 데이터를 실시간 암/복호화하므로 기존의 DES(Data Encryption Standard) 알고리즘을 적용한 연산 장치에 비해 더 높은 안전성을 제공한다.

;2> 셋째, 본 발명에 따른 암호 장치의 라인달 암/복호화용 라운드 연산 장치는 라운드 연산을 일정 수만큼 반복하게 하는 단순한 제어기의 추가로 블록 암호 데이터를 빠른 시간 안에 실시간으로 암호화/복호화할 수 있는 장점이 있다.

53> 넷째, 본 발명에 따른 암호 장치의 라운드 연산 장치는 기존의 128비트 단위 라운드 연산 장치에 비해서 절반에 가까운 적은 면적으로도 빠른 시간 내에 데이터를 암/복호화할 수 있다.

.54> 다섯째, 본 발명에 따른 암호 장치의 라운드 연산 장치는 적용 분야에 따라서 적절한 방법을 선택하여 구현할 수 있으며, 사용된 하드웨어 리소스의 양에 구애받지 않는 시스템에 적

용할 경우, 예컨대 64비트 단위의 라운드 처리가 아닌 128비트 단위의 라운드 처리 방식으로 응용하여 적용함으로써 두 배의 데이터 암호 복호 속도를 얻을 수 있다.

이상에서 설명한 것은 본 발명에 따른 라인달 블록 암호 장치와 그 암호화 및 복호화 방법을 실시하기 위한 하나의 실시예에 불과한 것으로서, 본 발명은 상기한 실시예에 한정되지 않고, 이하의 특허청구의 범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변경 실시가 가능한 범위까지 본 발명의 기술적 정신이 있다고 할 것이다.

【특허청구범위】

【청구항 1】

M비트의 입력데이터 및 N비트의 입력키를 가지며, 시프트_로우(Shift_Row), 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정을 포함하는 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 암호화하는 라인달 블록 암호화 장치에 있어서,

적어도 상기 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하는 라운드 연산 실행부와 상기 라운드 키 더하기 과정에 라운드키를 제공하기 위해 라운드키를 생성하는 라운드키생성부를 포함하는 라운드 연산부;

상기 라운드 연산부에 의해 수행되는 라운드 연산을 제어하는 라운드 연산 제어부; 및

상기 라운드 연산부에 의해 매 라운드의 중간단계에서 생성되는 M/m 비트의 중간 데이터 및 매 라운드의 종료단계에서 생성되는 M 비트의 데이터를 저장하기 위한 데이터기억부를 포함하는 것을 특징으로 하는 라인달 블록 암호화 장치.

【청구항 2】

제 1 항에 있어서,

상기 데이터기억부는 하나 이상의 레지스터를 포함하며, 그 크기의 전체합이 M(2m-1)/m 비트와 같거나 큰 것을 특징으로 하는 라인달 블록 암호화 장치.

【청구항 3】

M비트의 입력데이터 및 N비트의 입력키를 가지며, 역시프트_로우(Inverse Shift_Row), 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정을 포함하는 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 복호화하는 라인달 블록 복호화 장치에 있어서,

적어도 상기 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하는 라운드 연산 실행부와 상기 라운드 키 더하기 과정에 라운드키를 제공하기 위해 라운드키를 생성하는 라운드키생성부를 포함하는 라운드 연산부;

상기 라운드 연산부에 의해 수행되는 라운드 연산을 제어하는 라운드 연산 제어부; 및

상기 라운드 연산부에 의해 매 라운드의 중간단계에서 생성되는 M/m 비트의 중간 데이터 및 매 라운드의 종료단계에서 생성되는 M 비트의 데이터를 저장하기 위한 데이터기억부를 포함하는 것을 특징으로 하는 라인달 블록 복호화 장치.

【청구항 4】

제 3 항에 있어서,

상기 데이터기억부는 하나 이상의 레지스터를 포함하며, 그 크기의 전체합이 M(2m-1)/m 비트와 같거나 큰 것을 특징으로 하는 라인달 블록 복호화 장치.

【청구항 5】

M비트의 입력데이터 및 N비트의 입력키를 가지며, 시프트_로우(Shift_Row), 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정을 포함하는 암호화용 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 암호화하거나, 역시프트_로우(Inverse Shift_Row), 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정을 포함하는 복호화용 라운드 연산을 소정 회수만큼 반복하여 상기 M 비트의 입력데이터를 복호화하기 위한 라인달 블록 암복호화 장치에 있어서,

암호화 모드에서는, 적어도 상기 대치(Substitution), 믹스컬럼(MixColumn), 라운드 키 더하기(Add Round key)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하고, 복호화 모드에서는, 적어도 상기 역대치(Inverse Substitution), 라운드 키 더하기(Add Round key), 역믹스컬럼(Inverse MixColumn)의 변환과정에서는 데이터를 M/m 비트(m은 2, 3 또는 4) 단위로 처리하는 라운드 연산 실행부와 상기 라운드 키 더하기 과정에 라운드키를 제공하기 위해 라운드키를 생성하는 라운드키생성부를 포함하는 라운드 연산부;

상기 라운드 연산부에 의해 수행되는 라운드 연산을 제어하는 라운드 연산 제어부; 및

상기 라운드 연산부에 의해 매 라운드의 중간단계에서 생성되는 M/m 비트의 중간 데이터 및 매 라운드의 종료단계에서 생성되는 M 비트의 데이터를 저장하기 위한 데이터기억부를 포함하는 것을 특징으로 하는 라인달 블록 암복호화 장치.

【청구항 6】

제 5 항에 있어서, 상기 라운드 연산 실행부는,

상기 시프트_로우(Shift_Row)연산 및 역시프트_로우(Inverse Shift_Row)연산을 수행하는 시프트/역시프트_로우(Shift/Inverse Shift_Row)연산수단;

상기 대치(Substitution)연산 및 역대치(Inverse Substitution)연산을 수행하는 대치/역대치(Substitution/Inverse Substitution)연산수단;

상기 믹스컬럼(MixColumn)연산 및 역믹스컬럼(Inverse MixColumn)연산을 수행하는 믹스/역믹스컬럼(MixColumn/Inverse MixColumn)연산수단; 및

상기 라운드키 더하기(Add Round key)연산을 수행하는 라운드키 더하기(Add Round key) 연산수단을 포함하는 것을 특징으로 하는 라인달 블록 암복호화 장치.

【청구항 7】

제 6 항에 있어서, 상기 라운드 연산 실행부는,

암호 또는 복호모드를 나타내는 모드신호의 입력에 따라 각각 암호화용 라운드 연산 또는 복호화용 라운드 연산을 수행하도록; 상기 대치/역대치연산수단, 상기 믹스/역믹스컬럼연산수단 및 상기 라운드 키 더하기연산수단 사이에서 데이터의 흐름방향을 제어하는 복수의 역다중화수단을 포함하는 것을 특징으로 하는 라인달 블록 암복호화 장치.

【청구항 8】

제 5 항 내지 제 7항 중 어느 한 항에 있어서,

상기 데이터기억부는 하나 이상의 레지스터를 포함하며, 그 크기의 전체합이 M(2m-1)/m 비트와 같거나 큰 것을 특징으로 하는 라인달 블록 암복호화 장치.

【청구항 9】

M비트의 입력데이터와 N비트의 입력키를 입력받아, 소정 회수만큼 라운드연산을 수행하는 라인달 블록 암호화 방법에 있어서, 상기 방법은,

이전 라운드로부터의 M 비트 데이터를 시프트_로우(Shift_Row)변환을 가하고, 선택신호에 대응하는 M/m 비트(m은 2, 3 또는 4) 데이터만을 다음 단계로 출력하는 시프트_로우(Shift_Row)변환단계; 상기 M/m 비트 데이터에 대하여 대치(Substitution)변환을 실시하는 대치(Substitution)변환단계; 상기 M/m 비트 데이터에 대하여 믹스컬럼(MixColumn)변환을 실시하는 믹스컬럼(MixColumn)변환단계; 및 상기 M/m 비트 데이터에 동일한 크기의 라운드 키를 더하는 라운드 키 더하기(Add Round key)단계를 m개의 M/m비트 데이터 모두에 대해 수행하는 것을 하나의 라운드로 하는 라운드 연산단계; 및

상기 라운드키 더하기 단계에 라운드 키를 제공하기 위해 라운드키를 생성하는 라운드키 생성단계를 포함하는 것을 특징으로 하는 라인달 블록 암호화 방법.

【청구항 10】

제 9 항에 있어서,

상기 시프트_로우(Shift_Row)변환단계, 대치(Substitution)변환단계, 믹스컬럼(MixColumn)변환단계, 및 라운드 키 더하기(Add Round key)단계는 각각 M/m 비트 크기의 데이터를 처리할 수 있으며, 소정 타이밍에 따라 복수의 M/m 비트 데이터가 동일 시점에 상기 네

단계 중 복수의 단계를 각각 점유하여 처리될 수 있는 것을 특징으로 하는 라인달 블록 암호화 방법.

【청구항 11】

M비트의 입력데이터와 N비트의 입력키를 입력받아, 소정 회수만큼 라운드연산을 수행하는 라인달 블록 복호화 방법에 있어서, 상기 방법은,

이전 라운드로부터의 M 비트 데이터를 역시프트_로우(Inverse Shift_Row)변환을 가하고, 선택신호에 대응하는 M/m 비트(m은 2, 3 또는 4) 데이터만을 다음 단계로 출력하는 역시프트_로우(Inverse Shift_Row)변환단계; 상기 역시프트_로우 변환된 M/m 비트 데이터에 대하여 역대치(Inverse Substitution)변환을 실시하는 역대치(Inverse Substitution)변환단계; 상기 역대치 변환된 M/m 비트 데이터에 동일한 크기의 라운드 키를 더하는 라운드 키 더하기(Add Round key)단계; 및 상기 라운드 키를 더한 M/m 비트 데이터에 대하여 역믹스컬럼(Inverse MixColumn)변환을 실시하는 역믹스컬럼(Inverse MixColumn)변환단계를 m개의 M/m비트 데이터 모두에 대해 수행하는 것을 하나의 라운드로 하는 라운드 연산단계; 및

상기 라운드키 더하기 단계에 라운드 키를 제공하기 위해 라운드키를 생성하는 라운드키 생성단계를 포함하는 것을 특징으로 하는 라인달 블록 복호화 방법.

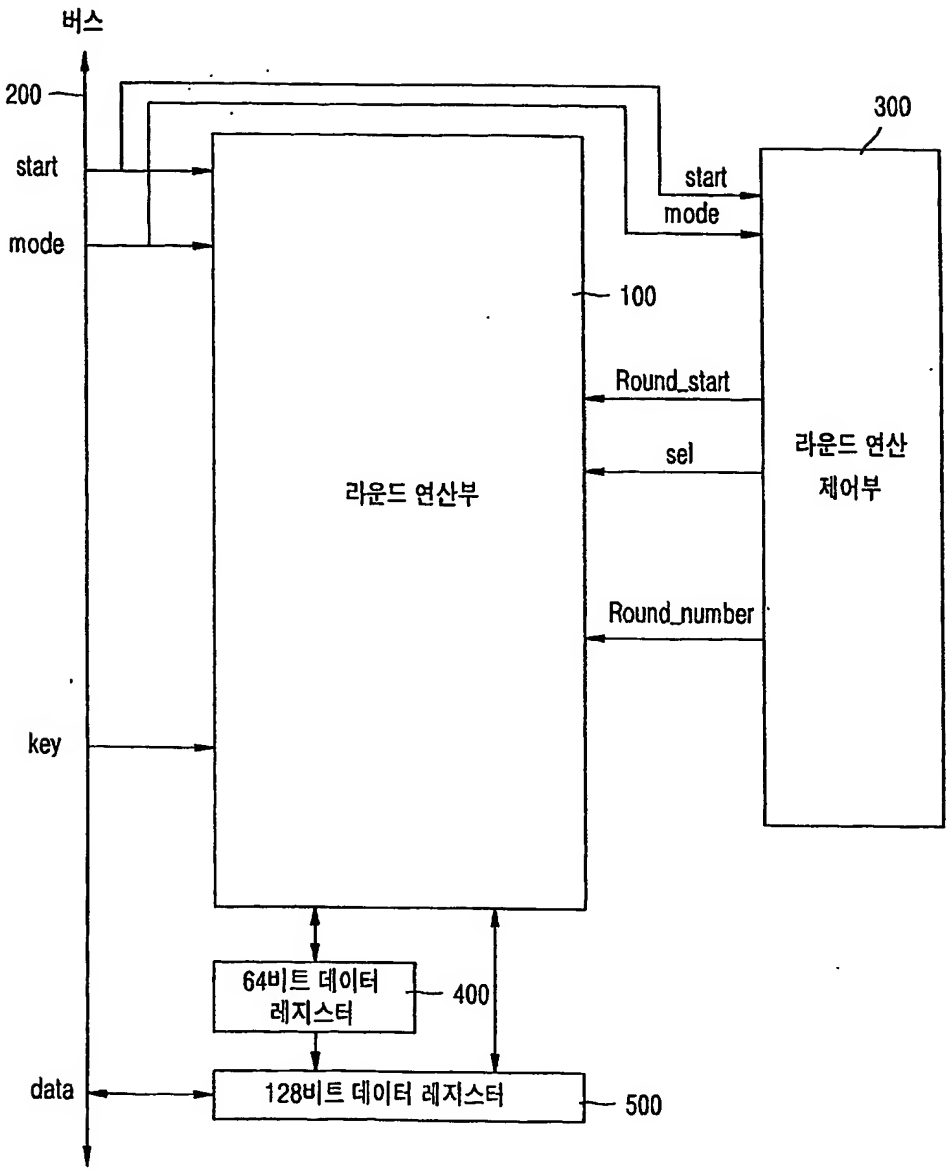
【청구항 12】

제 11 항에 있어서,

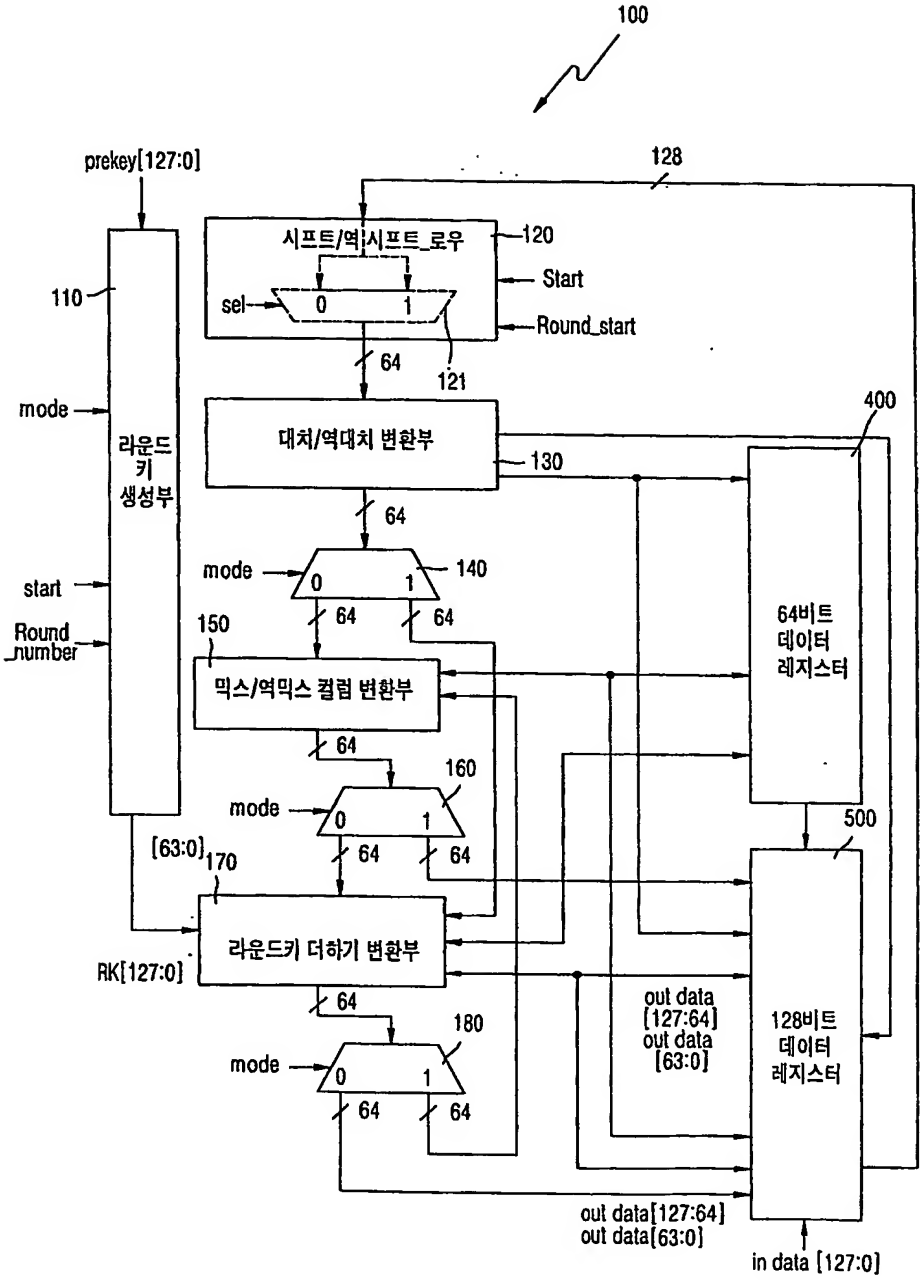
상기 역시프트_로우(Inverse Shift_Row)변환단계, 역대치(Inverse Substitution)변환단계, 라운드 키 더하기(Add Round key)단계, 및 역믹스컬럼(Inverse MixColumn)변환단계는 각각 M/m 비트 크기의 데이터를 처리할 수 있으며, 소정 타이밍에 따라 복수의 M/m 비트 데이터가 동일 시점에 상기 네 단계 중 복수의 단계를 각각 점유하여 처리될 수 있는 것을 특징으로 하는 라인달 블록 복호화 방법.

【도면】

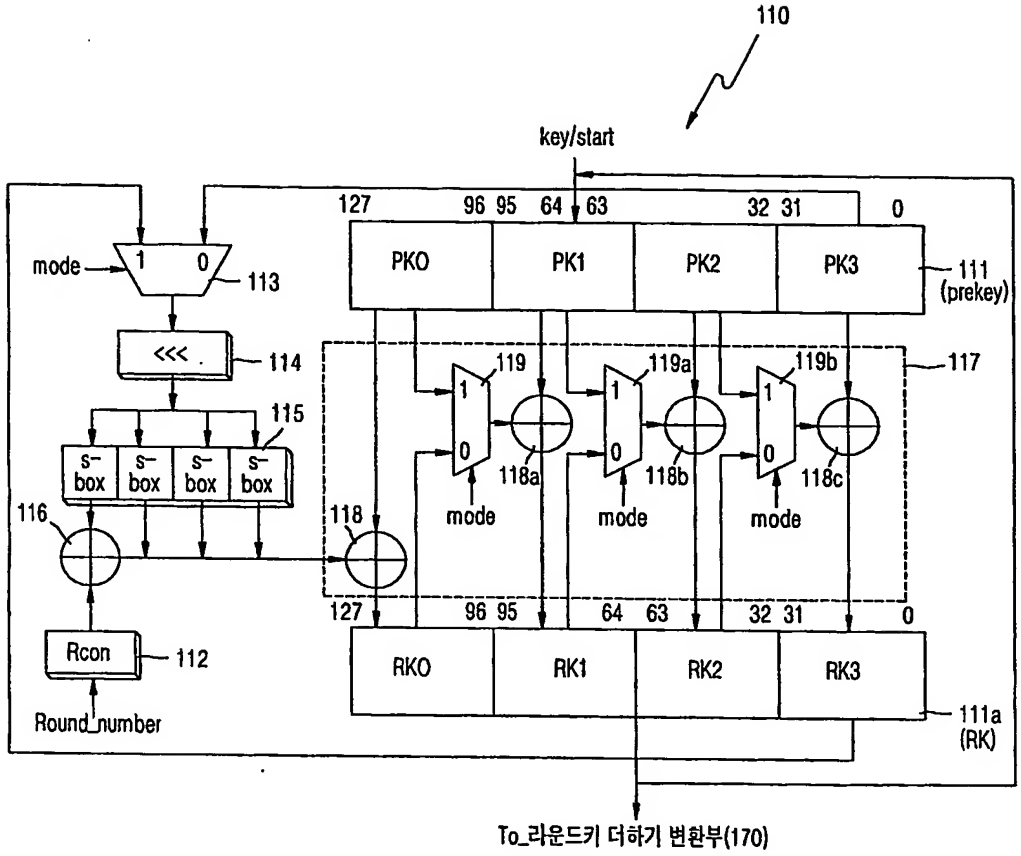
【도 1】


버스
200
start
mode
300
start
mode
100
Round_start
sel
라운드 연산부
라운드 연산 제어부
Round_number
key
64비트 데이터 레지스터
400
data
128비트 데이터 레지스터
500

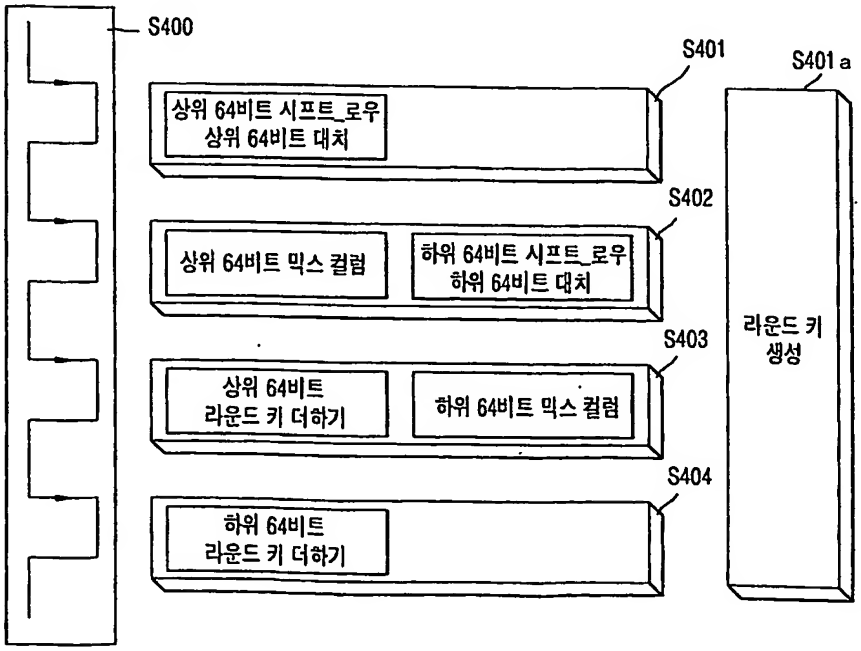
【도 2】


100
prekey[127:0]
128
시프트/역시프트_로우
120
Start
Round_start
sel
0
1
64
121
110
mode
라운드 키 생성부
start
Round number
대치/역대치 변환부
130
400
64
mode
140
150
믹스/역믹스 컬럼 변환부
160
170
[63:0]
라운드키 더하기 변환부
RK[127:0]
180
64비트 데이터 레지스터
500
128비트 데이터 레지스터
out data [127:64]
out data [63:0]
in data [127:0]

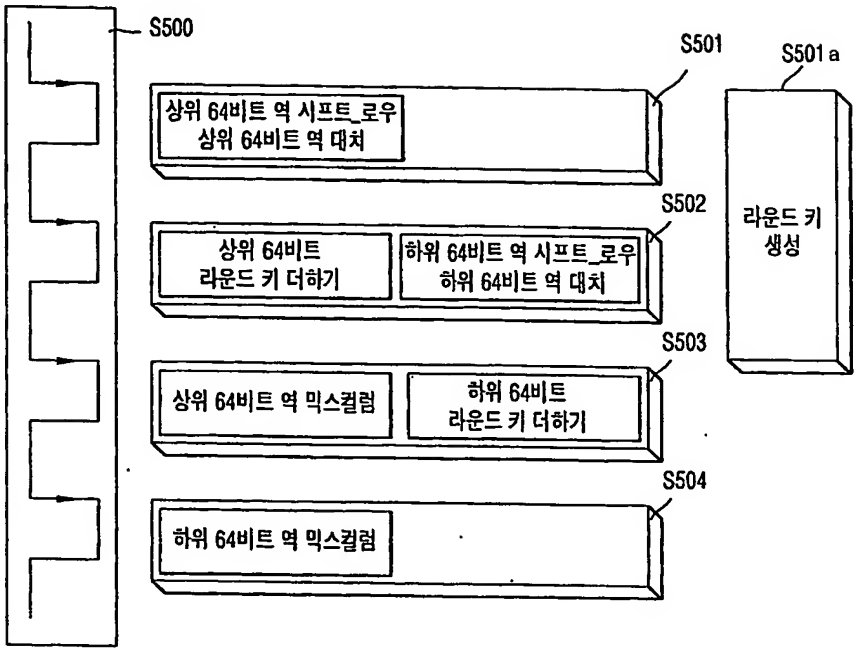
【도 3】


110
key/start
127
96 95
64 63
32 31
0
PK0
PK1
PK2
PK3
111
(prekey)
mode
1
0
113
<<<
114
115
S-box
S-box
S-box
S-box
116
118
119
119a
119b
117
118a
118b
118c
mode
mode
mode
Rcon
112
Round_number
RK0
RK1
RK2
RK3
111a
(RK)
To_라운드키 더하기 변환부(170)

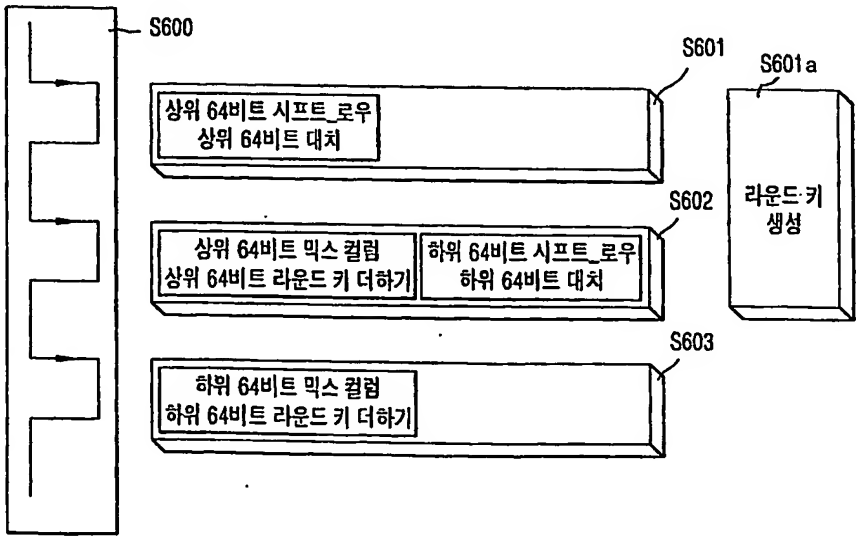
【도 4】


S400
S401
S401a
상위 64비트 시프트_로우
상위 64비트 대치
S402
상위 64비트 믹스 컬럼
하위 64비트 시프트_로우
하위 64비트 대치
S403
상위 64비트
라운드 키 더하기
하위 64비트 믹스 컬럼
S404
하위 64비트
라운드 키 더하기
라운드 키
생성

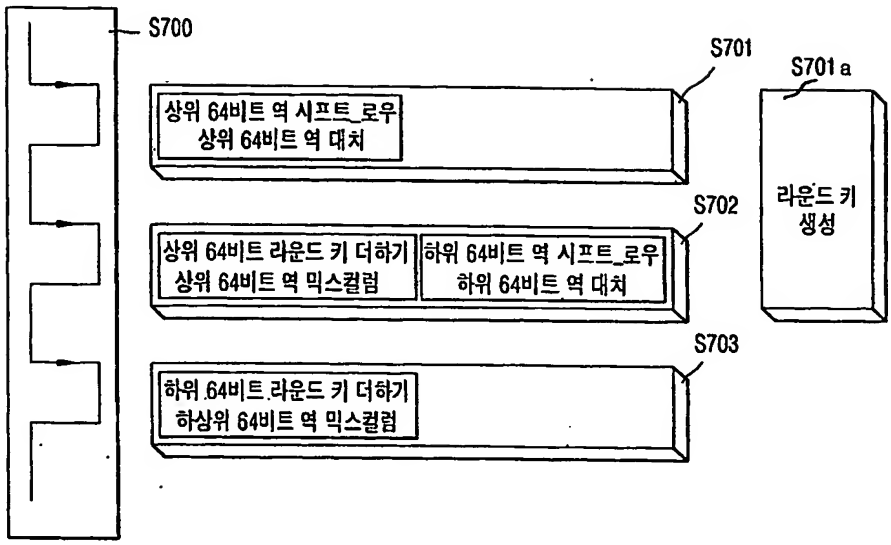
【도 5】


S500
S501
S501a
상위 64비트 역 시프트_로우
상위 64비트 역 대치
S502
상위 64비트
라운드 키 더하기
하위 64비트 역 시프트_로우
하위 64비트 역 대치
라운드 키
생성
S503
상위 64비트 역 믹스컬럼
하위 64비트
라운드 키 더하기
S504
하위 64비트 역 믹스컬럼

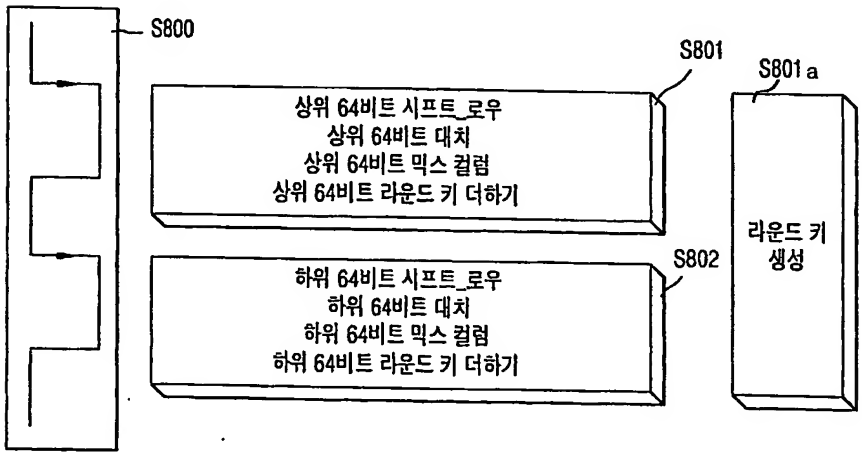
【도 6】


S600
S601
S601a
상위 64비트 시프트_로우
상위 64비트 대치
S602
상위 64비트 믹스 컬럼
상위 64비트 라운드 키 더하기
하위 64비트 시프트_로우
하위 64비트 대치
라운드 키
생성
S603
하위 64비트 믹스 컬럼
하위 64비트 라운드 키 더하기

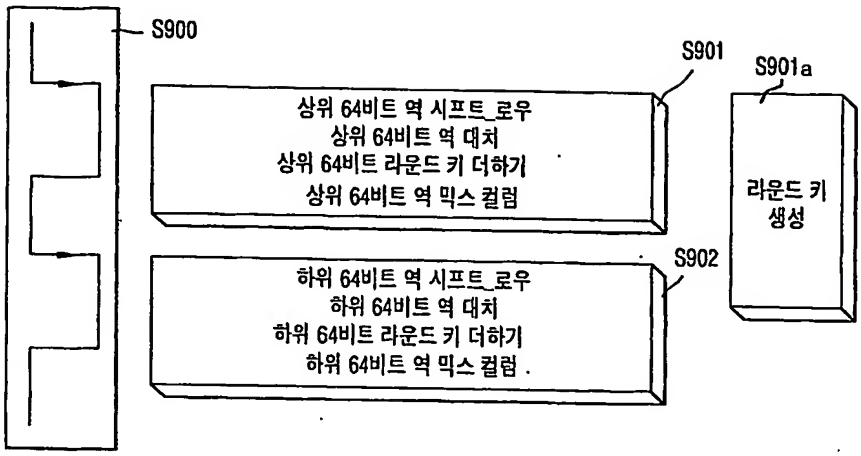
【도 7】


S700
S701
S701a
상위 64비트 역 시프트_로우
상위 64비트 역 대치
라운드 키
생성
S702
상위 64비트 라운드 키 더하기
상위 64비트 역 믹스컬럼
하위 64비트 역 시프트_로우
하위 64비트 역 대치
S703
하위 64비트 라운드 키 더하기
하상위 64비트 역 믹스컬럼

【도 8】


S800
S801
S801a
상위 64비트 시프트_로우
상위 64비트 대치
상위 64비트 믹스 컬럼
상위 64비트 라운드 키 더하기
라운드 키
생성
S802
하위 64비트 시프트_로우
하위 64비트 대치
하위 64비트 믹스 컬럼
하위 64비트 라운드 키 더하기

【도 9】


S900
S901
S901a
상위 64비트 역 시프트_로우
상위 64비트 역 대치
상위 64비트 라운드 키 더하기
상위 64비트 역 믹스 컬럼
라운드 키
생성
S902
하위 64비트 역 시프트_로우
하위 64비트 역 대치
하위 64비트 라운드 키 더하기
하위 64비트 역 믹스 컬럼